T.C.

ONDOKUZ MAYIS ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

KAFES TABANLI KRİPTOGRAFİK PROTOKOLLERİN VERİMLİ UYGULAMALARI

BİLGE KAĞAN YAZAR

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

SAMSUN

2020

TEZ ONAYI

Bilge Kağan YAZAR tarafından hazırlanan "**KAFES TABANLI KRİPTOGRAFİK PROTOKOLLERİN VERİMLİ UYGULAMALARI**" adlı tez çalışması 10/01/2020 tarihinde aşağıdaki jüri tarafından Ondokuz Mayıs Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans Tezi olarak kabul edilmiştir.

Danışman Dr. Öğr. Üyesi Erdem ALKIM
Bilgisayar Mühendisliği Anabilim Dalı

Jüri Üyeleri

Başkan Doç. Dr. Sedat Akleylek
Ondokuz Mayıs Üniversitesi
Bilgisayar Mühendisliği Anabilim Dalı

Üye Doç. Dr. Murat Cenk
Orta Doğu Teknik Üniversitesi
Uygulamalı Matematik Enstitüsü

Üye Dr. Öğr. Üyesi Erdem Alkım
Ondokuz Mayıs Üniversitesi
Bilgisayar Mühendisliği Anabilim Dalı

Yukarıdaki sonucu onaylarım. .../.../...

Prof. Dr. Bahtiyar Öztürk

Enstitü Müdürü

ETİK BEYAN

Ondokuz Mayıs Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez içindeki bütün bilgilerin doğru ve tam olduğunu, bilgilerin üretilmesi aşamasında bilimsel etiğe uygun davrandığımı, yararlandığım bütün kaynakları atıf yaparak belirttiğimi beyan ederim.

10/01/2020

Bilge Kağan YAZAR

ÖZET

Yüksek Lisans Tezi

KAFES TABANLI KRİPTOGRAFİK PROTOKOLLERİN VERİMLİ UYGULAMALARI

Bilge Kağan YAZAR

Ondokuz Mayıs Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Erdem ALKIM

Taraflar arasındaki iletişimin gizliliğinin ve bütünlüğünün korunabilmesi için kriptografik sistemlere ihtiyaç bulunmaktadır. Günümüzde güvenilir olduğu düşünülen açık anahtarlı şifreleme sistemlerinin çoğu çözümlerinin zor olduğu bilinen matematik problemlerine dayanmaktadır. Bu problemlerden en çok kullanılanları çarpanlara ayırma ve ayrık logaritma problemleridir. Günümüzde kullanılan bilgisayarlar ile bu problemlerin polinom zamanda çözülmesi çok zordur. 1994 yılında Peter Shor yaptığı bir çalışmada, çarpanlarına ayırma ve ayrık logaritma problemlerini kuantum bilgisayarlarla birlikte polinom zamanda çözebilen bir algoritma önermiştir. Bundan dolayı, günümüzde kullanılan RSA, Diffie-Hellman, ECC gibi açık anahtarlı şifeleme sistemlerinin, yeterli büyüklükte kuantum bilgisayarlar üretildiğinde güvensiz hale geleceği düşünülmektedir. Bu durum kuantum bilgisayarlar sonrasında kullanılabilecek güvenli kriptografik sistemlere ihtiyaç duyulduğunu göstermektedir. Kafes tabanlı sistemler kuantum sonrası için güvenilir yapılardan ve verimlilik açısından en çok tercih edilen sistemlerdendir.

Bu tez çalışmasında, kuantum bilgisayarlarla bile polinom zamanda çözülemeyeceği düşünülen, kafes yapısı üzerinde tanımlanmış zor problemlerden birisi olan hatalar ile öğrenme problemi (LWE) incelenmiştir. Bu problemi kullanan anahtar kapsülleme protokolleri ve bu protokollerde kullanılmakta olan matris vektör çarpım işlemleri incelenmiştir. Bu protokoller içerisinde kullanılmakta olan matris vektör çarpım işlemleri rastgele sayı üretiminden sonra en çok zaman almakta olan kısımlardır. Bu sebeple bu çarpım işlemlerinin hızlandırılması, optimize edilmesi ve verimli bir şekilde yapılmaları gerekmektedir. Tez kapsamında LWE problemini kullanan FrodoKEM, Lizard, Emblem ve Lotus protokolleri incelenmiştir. FrodoKEM protokolü içerisindeki matris vektör çarpım işlemleri temel alınarak verimli bir matris vektör çarpım kütüphanesi oluşturulmuştur ve diğer protokollere bu çarpım işlemleri uygulanarak elde edilen sonuçlar incelenmiştir. Ek olarak protokollerin genel verimliliklerini artırmak adına uygulamalarda yapılan değişikliklerden bahsedilmiştir.

Ocak 2020, 66 sayfa

Anahtar Kelimeler: Kuantum sonrası kriptografi, kafes tabanlı kriptografi, hatalar ile öğrenme problemi, anahtar kapsülleme protokolleri, matris vektör çarpımı.

ABSTRACT

Master's Thesis

EFFICIENT IMPLEMENTATIONS OF LATTICE BASED CRYPTOGRAPHIC PROTOCOLS

Bilge Kağan YAZAR

Ondokuz Mayis University
Graduate School of Sciences
Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Erdem ALKIM

Cryptographic systems are needed to protect the confidentiality and integrity of communication between the parties. Most public-key cryptosystems, which are now considered to be reliable, are based on mathematical problems that are known to be difficult to solve. The most commonly used problems are factorization and discrete logarithm problems. It is hard to solve these problems in polynomial time with the computers used today. In 1994, Peter Shor proposed an algorithm that could solve the problems of factorization and discrete logarithm with quantum computers in polynomial time. Therefore, public-key cryptosystems such as RSA, Diffie-Hellman, ECC, which are used today, are thought to become insecure when sufficient size quantum computers are produced. This shows that there is a need for secure cryptographic systems that can be used after quantum computers. Lattice-based systems are the most preferred systems in terms of efficiency and reliability for post-quantum structures.

In this thesis, learning with errors (LWE) problem is examined, which is one of the difficult problems defined on the lattice structure, which is thought to be solved in polynomial time even with quantum computers. Key encapsulation protocols using this problem and matrix-vector product used in these protocols are examined. The matrix-vector multiplications used in these protocols are the ones that take the most time after random number generation. For this reason, these multiplications need to be accelerated, optimized and efficiently performed. In this thesis, FrodoKEM, Lizard, Emblem and Lotus protocols using the LWE problem are examined. An efficient matrix-vector multiplication library was created based on the matrix-vector multiplication operations within FrodoKEM protocol and the results obtained by applying these multiplication operations to other protocols were examined. Also, changes in the implementations of the protocols to increase the overall efficiency of the protocols are mentioned.

January 2020, 66 pages

Key Words: Post quantum cryptography, lattice based cryptography, learning with errors problem, key encapsulation protocols, matrix vector product.

ÖNSÖZ VE TEŞEKKÜR

Akademik eğitim sürecimin bir sonraki aşaması olan yüksek lisans tez çalışmam boyunca yardım ve desteklerini esirgemeyen değerli hocalarıma teşekkür ederim.

Bugünlere gelmemde büyük emekleri olan, hiçbir fedakarlıktan kaçınmayan anneme ve babama sonsuz teşekkür ederim.

Çalışmalarım boyunca yardımlarını hiç esirgemeyen, bana destek olan çalışma arkadaşlarıma teşekkürü bir borç bilirim.

Bu tez çalışması EEEAG-116E279 numaralı proje kapsamında TÜBİTAK tarafından desteklenmiştir.

Ocak 2020, Samsun

Bilge Kağan YAZAR

İÇİNDEKİLER DİZİNİ

SİMGELER VE KISALTMALAR

KISALTMALAR

AES	Gelişmiş Şifreleme Standardı (Advanced Encryption Standard)
AVX	Vektör Komutları (Advance Vector Extensions)
BDD	Bounded Distance Decoding
BQP	Sınırlı-hata Kuantum Polinom-zamanı (Bounded-error Quantum Polynomial time)
CCA	Seçilmiş Şifremetin Saldırısı (Chosesn Ciphertext Attack)
CDF	Kümülatif Dağılım Fonksiyonu (Cumulative Distrubition Function)
CDT	Kümülatif Dağılım Tablosu (Cumulative Distrubution Table)
CPA	Seçilmiş Düzmetin Saldırısı (Chosen Plaintext Attack)
CVP	En Yakın Vektör Problemi (Closest Vector Problem)
DES	Veri Şifreleme Standardı (Data Encryption Standard)
DH	Diffie Hellman Anahtar Değişim Protokolü
ECC	Eliptik Eğri Kriptografi (Eliptical Curve Cryptography)
ECDH	Eliptik Eğri Diffie Hellman Anahtar Değişim Protokolü
GGH	Goldreich-Goldwasser-Halevi
IND – CCA	Seçilmiş Şifreli Metin Saldırısı Altında Fark Edilemezlik (Indistinguishability under chosen ciphertext attack)
IND – CPA	Seçilmiş Düz Metin Saldırısı Altında Fark Edilemezlik (Indistinguishability under chosen plaintext attack)
ISO	Uluslararası Standartlar Teşkilatı (International Organization for Standardization)
KEM	Anahtar Kapsülleme/Paketleme Protokolü (Key Encapsulation Mechanishm)
LPN	Gürültü ile Öğrenme (Learning Partiy with Noise)
LWE	Hatalar ile Öğrenme (Learning With Errors)
NIST	Amerikan Ulusal Standartlar ve Teknoloji Enstitüsü (National Institute of Standards and Technology)
NP	Polinom Zamanda Çözülemeyen (Non-polynomial-time)
NSA	Ulusal Güvenlik Ajansı (National Security Agency)
P	Polinom Zamanda Çözülebilen (Polynomial-time)
PKE	Açık Anahtarlı Şifreleme (Public Key Encryption)
RSA	Rivest–Shamir–Adleman
SHA	Güvenli Özet Algoritması (Secure Hash Algorithm)
SIS	Kısa Tam Sayı Çözümü (Short Interger Solution)
SSH	Secure Shell
SSL	Secure Socket Layer
SVP	En Kısa Vektör Problemi (Shorthest Vector Problem)

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Elektronik bir sistem tasarlanırken veya bir ağ üzerinde iletişim kurulurken, bilgi güvenliği kavramlarını uygulayabilmek için kriptografik sistemlere ihtiyaç duyulmaktadır. Kriptografi, günümüzde kullanılmakta olan bütün elektronik iletişim cihazları için büyük önem taşımaktadır. E–posta ve finansal işlemler yapılan sistemler gizlilik ve bütünlük gibi bilgi güvenliğinin temel gereklerini sağlamalıdır. Günümüzde kullanılan şifreleme sistemleri simetrik (gizli anahtarlı) ve asimetrik (açık anahtarlı) olmak üzere ikiye ayrılmaktadır. Gizli anahtarlı sistemlerde şifreleme ve şifre çözme işlemleri yapılırken iki işlem içinde bir tane gizli anahtar kullanılmaktadır. Bu sistemler pratikte çok hızlı ve kolay hesaplanabilir sistemlerdir. Günlük hayatta standartlarda belirtilen simetrik şifreleme sistemlerine örnekler 3DES ve AES'tir (FIPS 197, 2001). Fakat büyük sistemler tasarlanmak istendiğinde, bu şifreleme şemaları kullanılacağı zaman gizli anahtarın diğer kullanıcılar ile paylaşılması problemi ortaya çıkmaktadır. Bu duruma çözüm olarak Diffie ve Hellman tarafından açık anahtarlı şifreleme sistemi önerilmiştir (Diffie ve Hellman, 1976). Açık anahtarlı sistemler hesaplandıktan sonra geri döndürülmesi çok zor olan tek yönlü fonksiyonları temel almaktadır. Diffie-Hellman açık anahtarlı şifrelemesinin güvenliği, günümüz bilgisayarlarına göre çözülmesi çok zor bir problem olan ayrık logaritma problemine dayanmaktadır. Açık anahtarlı sistemlerde tek bir anahtar yerine biri açık biri gizli anahtar olmak üzere iki anahtar bulunmaktadır ve bu anahtarlar matematiksel olarak birbirine bağlıdır. Bu sistemlerde gizli anahtara erişim sadece alıcı tarafından, açık anahtara erişim ise alıcıya veri göndermek isteyen herkes tarafından sağlanabilir. Basit olarak; alıcıya şifrelenmiş bir veri göndermek için veri alıcının açık anahtarı ile şifrelenir. Açık ve gizli anahtar ikilisi birbirine matematiksel olarak bağlı olduğu için açık anahtarı ile şifrelenmiş bir veriyi alıcı sadece kendi gizli anahtarı ile açabilir. Böylelikle anahtar değişimi durumu ortadan kalkıp daha güvenilir sistemlerin oluşmasına yol açılmıştır. Açık anahtarlı kriptografinin uygulama alanları; şifreleme, dijital/elektronik/sayısal imzalama ve anahtar değişimi olarak üç ana başlık altında ifade edilebilir. Açık anahtarlı şifreleme sistemlerinden olan RSA günümüzde en çok bilinen ve kullanılan açık anahtarlı şifreleme sistemidir denilebilir. Bu sistem 1978 yılında Rivest, Shamir

ve Adlemann tarafından önerilmiştir. RSA algoritması temelde büyük sayıları çarpanlara ayırma problemine dayanmaktadır ve bu problemin çözümünün zorluğundan dolayı günümüzde halen kullanılmakta olan bir sistemdir. 1985 yılında El-Gamal tarafından önerilen bir diğer sistem dairesel gruplardaki ayrık logaritma probleminin zorluğuna dayanan bir şifreleme sistemidir (ElGamal, 1985). Devamında eliptik eğrilerin kriptografide kullanılabileceği Miller ve Koblitz tarafından keşfedilmiştir (Miller, 1985; Koblitz, 1987). Bu sürecin devamında çeşitli açık anahtarlı kriptosistemler önerilmiş ve bu alan gelişim göstermeye devam etmiştir.

NIST, NSA ve ISO gibi kurumlar tarafından standart olarak kabul edilmiş bu şifreleme sitemleri yeterli güvenliği sağlayabilecek parametreler ile kullanıldığı zaman günümüzde kullanılan bilgisayarlardan gelecek ataklara karşı güvenilir oldukları düşünülmektedir. Örnek olarak; internette taraflar arası güvenli şekilde iletişim kurmak için kullanılan SSH ve SSL gibi güvenlik protokollerinde anahtar değişim protokolü olarak DH ve ECDH sistemleri kullanılmaktadır. Bu sistemlere ek olarak RSA sistemi ile belirlenen gizli anahtar karşı tarafın açık anahtarı ile şifrelenip gönderilmektedir. RSA sistemi kullanılarak da anahtar değişimi yapılabilmektedir.

İlerleyen zamanlarda, yeterince büyük ölçekte kuantum bilgisayarlar üretilebilirse, günümüzde kullanılmakta olan RSA, DH, ECC gibi sistemlerin kullanmakta olduğu zor problemlerin kolayca çözülebilir hale geleceği düşünülmektedir. Bundan dolayı kuantum bilgisayarlar güvenli ve doğrulanmış iletişimi tehdit eden bir durum olarak karşımıza çıkmaktadır. Çünkü kuantum bilgisayarlar, belirli zor problemleri günümüz bilgisayarlarına oranla çok daha hızlı çözebilmektedir. Buradan kuantum bilgisayarların tüm gizli anahtarları ayrıntılı ve hızlı bir şekilde arayabileceği, gizli anahtarı klasik bilgisayarlara oranla daha çabuk bulabileceği anlaşılabilir.

Peter Shor 1994 yılında yaptığı bir çalışmada, büyük tam sayıların çarpanlara ayrılmasının kuantum bilgisayarla birlikte temelde değişeceğini söylemiştir. Shor, açık anahtarlı kriptografide en çok kullanılan problemlerden ikisi olan çarpanlara ayırma ve ayrık logaritma problemlerinin kuantum bilgisayarlarla birlikte polinom zamanda çözülebileceğini göstermiştir (Shor, 1994).

1996 yılında Grover düzenlenmemiş veri tabanlarını aramak için kuantum bilgisayarları kullanan bir algoritma önermiştir. Bu algoritma ile, içinde N tane veri bulunan düzenlenmemiş bir veri tabanı içerisinde istediğimiz değeri $\sqrt{N}$ adımda bulabileceğimizi göstermiştir. Günümüz bilgisayarlarında ise N/2 aramada istediğimiz değeri bulabilmekteyiz (Mavroeidis vd, 2018). Bone ve Castro 1997 yılında Grover algoritmasının yapılacak bir uygulamasının DES sistemini güvensiz hale getireceğini göstermiştir. 56 bit güvenliğe sahip DES algoritmasının gizli anahtarının 185 adımda elde edilebileceğini söylemişlerdir (Bone ve Castro, 1997).

Sonuç olarak Shor'un algoritmasının asimetrik şifreli sistemleri, Grover'ın algoritmasının düzgün parametreler ile kullanılmazsa simetrik şifreli sistemleri kuantum sonrası kriptografi için güvensiz bıraktığı söylenebilir. Ek olarak Bernstein ve arkadaşları yaptıkları bir çalışmada; Grover algoritmasının simetrik şifreleme sistemleri üzerine bazı uygulamaları olduğunu ancak bu uygulamaların Shor algoritmasının uygulamaları kadar hızlı olmadığını söylemişlerdir (Bernstein vd, 2009).

Hali hazırda kullanılan simetrik şifreleme sistemlerin güvenliği sağlamaya devam etmesi için gizli anahtarlarda kullanılan bit sayısının artırılması yöntemi tercih edilmektedir. Bu da bir anahtarı elde etmek için gerekli arama sayısını artırmaktadır. Günümüzde kullanılan asimetrik şifreleme sistemleri olan RSA, ECC, DH, ECDH sistemlerinin Shor algoritmasından sonra kuantum bilgisayarlara karşı tamamen güvensiz durumda oldukları düşünülmektedir. Simetrik şifreli sistemler ise Grover algoritmasından sonra düzgün parametreler ile kullanılmazlar ise güvensiz durumdadırlar. Örneğin AES sistemi 128-bit ve üzeri anahtar ile kullanılmadığı zaman kuantum bilgisayarlara karşı güvenli olmadığı düşünülmektedir.

Burada bahsedilmesi gereken diğer bir kriptografik yapı özet fonksiyonlardır. Özet fonksiyonları güvenlikleri sabit bir çıktı uzunluğuna bağlı yapılardır. Bu durumdan dolayı simetrik şifreli sistemler ile benzer bir probleme sahiplerdir. Brassard ve arkadaşları yaptıkları bir çalışmada Grover algoritmasının doğum günü paradoksu ile birleştirilebileceğini göstermişlerdir (Brassard vd, 1998). $\sqrt[3]{N}$ boyutlu bir tablo ve Grover algoritması ile özet fonksiyonlarda çakışma bulunabilmektedir. Grover algoritmasına karşı x-bitlik güvenli bir sistem için 3x-bitlik bir çıktı gerekmektedir (Mavroedis vd, 2018).

Günümüzde kullanılmakta olan şifreleme sistemlerinin bazılarının klasik bilgisayarlar ve kuantum bilgisayarlara karşı sağladığı güvenlik seviyeleri Çizelge 1.1'de gösterilmiştir.

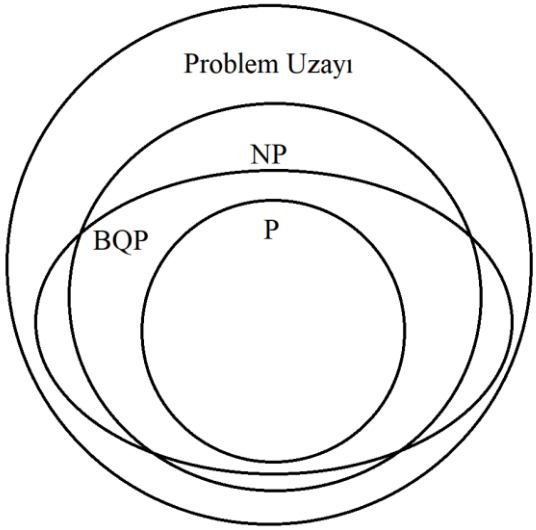
Çizelge 1.1. Günümüzde kullanılan bazı kriptosistemlerin klasik bilgisayar ve kuantum sonrası güvenlik seviyeleri (Mavroeidis vd, 2018)

Sistem	Anahtar Boyutu	Günümüz Güvenlik Seviyesi	Kuantum Sonrası Güvenlik Seviyesi
RSA – 1024	1024 – bit	80 – bit	0 – bit
RSA – 2048	2048 – bit	112 – bit	0 – bit
ECC – 256	256 – bit	128 – bit	0 – bit
ECC – 384	384 – bit	256 – bit	0 – bit
AES – 128	128 – bit	128 – bit	64 – bit
AES – 256	256 – bit	256 – bit	128 – bit

Yukarıda anlatılan durumlardan anlaşıldığı üzere; kuantum sonrası için güvenli açık anahtarlı sistemlere büyük ölçüde ihtiyaç duyulmaktadır. NIST 2016 yılında yayınladığı bir raporda; yeterince büyük ölçekte kuantum bilgisayarlar üretildiği takdirde, şu anda kullanılmakta olan şifreleme sistemlerinin kuantum bilgisayarlardan gelecek ataklara karşı güvenli olmayacağını söylemiştir (Chen vd, 2016).

Literatürde tasarlanmış olan kriptografik sistemlerin hepsinin alt yapısında bilgisayar bilimlerindeki çözümlerinin zor olduğu bilinen matematiksel problemler kullanılmaktadır. Bu problemler için şu anda tanımlanmış olan iki temel sınıf P ve NP sınıflarıdır. P sınıfı; kullanılan problemin klasik bilgisayarlar tarafından polinom zamanda çözülebileceği, NP sınıfı ise problemin klasik bilgisayarlar tarafından polinom zamanda çözülemeyeceği (varsayımsal yaklaşımlar hariç) düşünülen problemlerdir. Fakat NP sınıfında olan bazı zor problemler kuantum bilgisayarlar ile birlikte polinom zamanda çözülebilir duruma geldiğinden 1997 yılında Bernstein ve Vazirani tarafından yapılan bir çalışmada tanımlanmış olan BQP adlı yeni bir sınıf

içine dahil edilmektedir. Bu durumdan dolayı kuantum bilgisayarlara karşı yine NP sınıfında bulunan problemlere dayalı sistemler üretilmesi gerekmektedir (Chen vd, 2016). Fakat BQP sınıfının NP sınıfı ile ilişkisi henüz tam olarak belirlenebilmiş bir durum değildir (Nielsen ve Chuang, 2000). BQP sınıfının P sınıfı ve problem uzayı arasında bir yerde olduğu düşünülmektedir.



Şekil 1.1. Kuantum bilgisayarlar sonrası temsili olarak kullanılan problem sınıfları (Nielsen ve Chuang, 2000)

Tam bu kısımda kuantum sonrası kriptografi kavramı kendini göstermektedir. Kuantum sonrası kriptografinin amacı; günümüz iletişim, ağ protokolleri, bilgisayarlarında ve kuantum bilgisayarlarda verimli şekilde çalışan şifreleme / kapsülleme, anahtar değişim, dijital imza sitemleri geliştirmektir (Chen vd, 2016; Mavroeidis vd, 2018). Kuantum sonrası kriptografi için güvenli olduğu düşünülen sistemler; çok değişkenli polinom tabanlı, kafes tabanlı, kod tabanlı, özet tabanlı ve izojeni tabanlı sistemler olmak üzere beş ana sınıf altında toplanmaktadır.

Çok değişkenli polinom tabanlı sistemler; çok değişkenli polinom sistemlerini sonlu cisimler üzerinde çözmenin zorluğuna dayanan sistemlerdir (Ding ve Yang, 2009). Bu yöntem şifreleme ve dijital imza şemaları için kullanılabilmektedir. Fakat çok değişkenli polinomları temel alarak oluşturulmak istenen bir çok şifreleme sistemi başarısız olmuştur (Buchanan ve Woodward, 2017). Ancak Rainbow sistemi çok değişkenli polinomları kullanan ve kuantum dayanıklı bir imzalama şeması olarak öne çıkmaktadır (Ding ve Schmidt, 2005).

Kafes tabanlı sistemler; kafes yapısı üzerindeki problemlerin çözümlerinin zorluğuna dayanarak önerilmiş olan açık anahtarlı şemalarıdır. Diğer sistemlere oranla düşük boyutlu anahtar kullanması ve en kötü durumda bile güvenliği garanti eden problemleri ile kuantum sonrası kriptografi için en önemli adaylardır.

Kod tabanlı sistemler; McEliece kriptosisteminin (McEliece, 1978) önerilmesiyle ortaya çıkmış bir alandır. Bu sistemlerde temel olarak hata düzeltme kodlarında kullanılmakta olan doğrusal kodlar ve matris vektör çarpımı kullanılmaktadır (Bernstein vd, 2009).

Özet tabanlı sistemler; özet yöntemleri kullanılarak dijital imzalar oluşturmakta kullanılan sistemlerdir. Bu sistemlerin tek dezavantajı, tek seferlik imza yöntemlerinin üzerine inşa edilmiş olmalarıdır. Bu yüzden üretilecek imza sayısı sınırlıdır (Bernstein vd, 2009).

İzojeni tabanlı sistemler; eliptik eğrilerin özelliklerine ve aynı sayıda noktaya sahip iki eğri arasında bir izojeni oluşturma zorluğuna dayanan sistemlerdir (Buchanan ve Woodward, 2017). İlk olarak 1999 yılında Galbraith tarafından ele alınmıştır ve diğer alanlara nispeten yeni bir araştırma alanıdır.

Bu tez kapsamında kuantum sonrası için güvenli olduğu düşünülen kafes tabanlı anahtar kapsülleme protokolleri ve bunların verimli uygulamaları üzerine çalışılmıştır. Kafes yapısı üzerinde SVP, CVP, LWE, SIS gibi tanımlanmış olan bir çok zor problem bulunmaktadır. Tez kapsamında LWE problemi ve güvenliği LWE probleminin zorluğuna dayanmakta olan FrodoKEM, Lizard, Emblem ve Lotus anahtar kapsülleme protokolleri incelenmiş ve FrodoKEM protokolü içersindeki matris vektör çarpım işlemlerine odaklanılmıştır. Kafes tabanlı sistemler veya içerisinde herhangi bir matris vektör çarpım işlemi bulunduran başka sistemlerde kullanılması için açık kaynaklı olarak verimli bir matris vektör çarpımı kütüphanesi yapılması amaçlanmıştır. Devamında FrodoKEM dışındaki protokollerin FrodoKEM benzeri verimli uygulamaları yapılabilmesi amacı ile FrodoKEM protokolü temel alınarak, bütün protokollerin içerisinde bulunacağı sadece parametre seçerek çalışacak bir sistem tasarlanılması hedeflenmiştir.

1.1. Önceki Çalışmalar

Kafeslerin kriptografik sistemler oluşturulmasında kullanılabileceği ilk olarak 1996 yılında Ajtai tarafından ortaya atılmıştır. Ajtai en kötü durumda dahi güvenliği garanti eden bir kafes tabanlı şifreleme sistemi önermiştir (Ajtai, 1996). Devamında Hoffstein 1996 yılında yaptığı bir çalışmada NTRU sistemini önermiştir. Bu sistem şifreleme ve dijital imza oluşturmak için kullanılabilen bir sistemdir. NTRU sistemi polinomların çarpanlara ayrılmasının zorluğuna dayanan ve kuantum sonrası için 128 bit güvenlik sağlayarak Shor algoritmasına karşı güvenli olduğu düşünülen bir sistemdir (Hirschhorn vd, 2009). 1997 yılında Ajtai ve Dwork yaptıkları bir çalışmada; SVP'nin en kötü durum ve ortalama durum karmaşıklığı arasındaki ilişkiyi göstererek bir şifreleme sistemi önermişlerdir. Önerdikleri sistemin en kötü durumda dahi kanıtlanabilir güvenlik seviyesine sahip olduğunu iddia etmişlerdir (Ajtai ve Dwork, 1997). Ancak 1998 yılında Nguyen ve Stern yaptıkları bir çalışmada bunun tersini göstermişlerdir (Nguyen ve Stern, 1998). 1997 yılında NP sınıfında bir problem olan CVP problemini kullanan ve şifre çözme hatalarını ortadan kaldıran bir sistem olan GGH sistemi önerilmiştir (Goldreich vd, 1997). GGH sistemi Ajtai – Dwork sisteminden daha kullanışlı olmasına rağmen Nguyen 1999 yılında yaptığı bir çalışmada, CVP örnekleri üzerinden düz metinler (plain text) hakkında az da olsa bilgi edinilebildiğini göstermiştir (Nguyen, 1999).

2005 yılında Oded Regev LWE problemini ve bu problemi temel alan bir açık anahtarlı şifreleme sistemi önermiştir (Regev, 2005). 2011 yılında Lindner ve Peikert tarafından yapılan bir çalışmada; LWE problemi içinde dikdörtgen şeklinde matrisler yerine kare matris kullanmak gibi birçok değişiklik yaparak daha verimli bir LWE tabanlı sistem önerilmiştir (Lindner ve Peikert, 2011). Bu çalışmalardan sonra, Regev ve Lindner-Peikert şifreleme sistemlerini temel alan bir sistem olan Frodo (Bos vd, 2016) anahtar değişim protokolü önerilmiştir. 2017 yılında Frodo protokolünü temel alan, NIST' in çağrısı sonrası, standart LWE problemini kullanan ve bu tezin ana kapsamını oluşturan FrodoKEM (Naehrig vd, 2017) anahtar kapsülleme protokolü önerilmiştir. FrodoKEM protokolünden farklı olarak NIST'e önerilmiş olan standart kafesleri kullanan diğer anahtar kapsülleme protokolleri; Lizard (Cheon vd, 2017), Emblem (Seo vd, 2017), Lotus (Le Trieu Phong vd, 2017) protokolleridir.

1.2. Motivasyon ve Katkı

Kriptografik sistemlere yapılan ataklardan şifrelenmiş veriler elde edilebilmektedir. Kuantum bilgisayarlar klasik bilgisayarlara oranla çok daha fazla işlem gücüne sahip olmaları nedeniyle, kuantum bilgisayarlara karşı güvenli sistemlerin üretilmesi için kuantum bilgisayarların tam anlamıyla günlük hayata girmesinin beklenmemesinin gerektiği düşünülmektedir (Chen vd, 2016). Günümüzde kullanılan şifreleme sistemleri tam olarak güvensiz duruma geldiklerinde, bu sistemlerin kaldırılıp yerlerine güvenli oldukları düşünülen sistemlerin getirilmesi gerekmektedir. Fakat bunun yapılması oldukça uzun bir süreçtir. Bu nedenle NIST kuantum bilgisayarlara karşı güvenli sistemlerin geliştirilmesi için bir çağrıda bulunmuştur. NIST 2016 yılında yayınladığı raporda kuantum sonrası için güvenlik seviyelerini belirlemiş ve 2017 yılında kuantum sonrası güvenli yeni sistemler oluşturulması için bir standartlaştırma projesi başlatmıştır.

Çizelge 1.2. Kuantum sonrası kriptografi için NIST' in önerdiği güvenlik seviyeleri

Güvenlik Seviyesi	Kategori
AES – 128	Seviye 1
SHA – 256	Seviye 2
AES – 192	Seviye 3
SHA – 384	Seviye 4
AES – 256	Seviye 5

Bu projenin amacı AES veya SHA gibi tek bir standart oluşturmak değil, kuantum sonrası için güvenli birçok sistem oluşturulmasıdır. Bu doğrultuda çeşitli protokoller NIST'e önerilmiştir ve ilk tur için incelemeleri yapılmıştır. Düzenlenen projenin ilk turunda toplamda 82 tane sistem önerilmiştir. Bu sistemlerin sınıflarıyla birlikte sayıları Çizelge 1.3'de verilmiştir.

Çizelge 1.3. Kuantum sonrası kriptografi NIST projesi ilk turunda önerilmiş olan sistemlerin sınıflarıyla birlikte sayıları (Moody, 2017; NIST, 2018)

Protokol Türü	İmzalama	Şifreleme / Anahatar Kapsülleme	Toplam
Kafes Tabanlı	4	24	28
Kod Tabanlı	5	19	24
Çok Değişkenli Polinom Tabanlı	7	6	13
Özet Tabanlı	4	-	4
Diğer	3	10	13
Toplam	23	59	82

NIST standartlaştırma projesi halen devam etmekte olan bir projedir. Tez kapsamında ise sadece standart LWE problemini temel alan anahtar kapsülleme protokolleri incelenmiştir. Standartlaştırma projesinde önerilmiş olan diğer protokoller hakkında güncel ve ayrıntılı bilgiye SafeCrypto'dan (SafeCrypto, 2019) ulaşılabilir.

Son zamanlarda kafes tabanlı kriptografik sistemler kuantum sonrası kriptografi alanında büyük oranda ilgi çekmektedir ve kafes tabanlı kriptosistemlerin kuantum sonrası güvenliği sağlamak için en büyük adaylar oldukları düşünülmektedir (Bernstein vd, 2009). Bunun sebeplerinden bazıları; kafes tabanlı sistemlerin en kötü durumda dahi kanıtlanabilir güvenlik seviyesi sağlaması ve diğer sistemlere oranla küçük boyutlu anahtarlar kullanmasıdır. Çizelge 1.3'de görüldüğü üzere; NIST'e ilk turda önerilen sistemler arasında en çok kafes tabanlı sistemler bulunmaktadır. Önerilen kafes tabanlı sistemlerin yaklaşık %85 i şifreleme / anahtar kapsülleme sistemleridir. Standartlaştırma sürecinin ilk turunda önerilen sistemlerin incelemeleri yapılmış ve devamında 2019 yılında NIST ikinci tura geçen 26 protokolü açıklamıştır (Alagic vd, 2019). Bu protokoller Çizelge 1.4'de verilmiştir. Kafes tabanlı sistemlerin ikinci turda da çoğunlukta olduğu görülmektedir.

Çizelge 1.4. Kuantum sonrası kriptografi NIST projesi ikinci tura geçen protokoller ve sınıfları

Protokol Türü	İmzalama	Şifreleme / Anahatar Kapsülleme
Kafes Tabanlı	qTesla, CRYSTALS – DILITHIUM, FALCON	FrodoKEM, NTRU Prime, Three Bears, CRYSTALS – KYBER, NTRU, SABER, Round5, New Hope, LAC
Kod Tabanlı	-	BIKE, Classic McEliece, ROLLO, RQC, HQC, LEDACrypt, NTS – KEM
Çok Değişkenli Polinom Tabanlı	LUOV, GeMSS, MQDSS, Rainbow	-
Özet Tabanlı	SPHINCS+	-
Diğer	Picnic	SIKE

Önerilmiş olan sistemlerin içerisinde LWE problemini temel alan şifreleme / anahtar kapsülleme şemaları bulunmaktadır. Oluşturulan kriptografik sistemlerde LWE probleminin kullanılmasının sebebi; bu problemin kuantum bilgisayarlar dahil polinom zamanda çözülemeyeceğinin düşünülmesi ve kriptografik sistemler oluşturulurken verimli bir şekilde uygulanabilir olmasıdır. Tez kapsamında kafes yapısı üzerinde tanımlı zor problemlerden biri olan LWE problemi ve standart LWE problemini kullanan bir protokol olan FrodoKEM başta olmak üzere Lizard, Emblem ve Lotus protokolleri incelenmiştir.

LWE tabanlı sistemlerde; LWE problemi tanımı gereği matris vektör çarpım işlemleri bulunmaktadır. Matris vektör çarpımı bu problemi kullanan sistemlerde rastgele sayı üretiminden sonra en çok zaman almakta olan kısımdır.

n × n boyutunundaki iki matrisi çarpma (matris-vektör) işlemi için çok sayıda yöntem bulunmaktadır. İki matrisi çarpmak için kullanılan standart matris çarpım (schoolbook) yönteminin karmaşıklığı $O(n^3)$'dür. Böl-ve-fethet mantığına dayanan Strassen yönteminde ise karmaşıklık $O(n^{2.81})$'dir. Kare olmayan matrisler için Strassen yönteminin doğrudan uygulanabilmesi mümkün değildir.

İncelenen protokoller içerisinde FrodoKEM protokolü dışında matris çarpım işlemleri standart matris çarpım yöntemi kullanılarak yapılmaktadır. Kullanılan matrislerin boyutları çok büyük olduğundan bu yöntem çok fazla zaman harcamaktadır. Bu sebeple bu çarpımların hızlandırılması, optimize edilmesi ve kabul edilebilir bir zaman içerisinde yapılması gerekmektedir. Bu durumdan dolayı tez kapsamında FrodoKEM içerisindeki matris vektör çarpım işlemleri temel alınarak, kafes tabanlı bir sistem oluşturmak isteyen kişilerin kullanabilmesi amacıyla açık kaynaklı olarak bir matris vektör çarpım kütüphanesi yapılması amaçlanmıştır. Bunun için FrodoKEM protokolündeki çarpım işlemleri matrisin şekli fark etmeksizin çalışabilecek duruma getirilip genelleştirilmiştir. Ek olarak bu matris çarpım kütüphanesi içerisindeki çarpımların Lizard, Emblem ve Lotus protokollerine uygulanıp protokollerdeki değişimlerin incelenmesi amaçlanmıştır.

Çizelge 1.4'de görüldüğü üzere tez kapsamında incelenen protokollerden FrodoKEM dışındaki diğer protokoller NIST projesinde ikinci tura geçememişlerdir. Bu durumdan dolayı ikinci tura geçemeyen Lizard, Emblem ve Lotus protokollerinin FrodoKEM benzeri daha verimli uygulamalarının yapılması amaçlanmıştır.

Son olarak incelenen bu dört protokolün hepsinin tek bir çatı altında toplanması amaçlanmıştır. Bu işlemin amacı; standart LWE problemini kullanarak kriptografik bir sistem üretmek isteyen kişilerin istediği yöntemleri sadece parametre seçerek kullanıp kendi sistemini otomatik bir şekilde üretmesidir.

1.3. Organizasyon

Bölüm 2'de tez kapsamına incelenen protokollerin daha iyi anlaşılabilmesi için bazı matematiksel ve kriptografik terimlerden bahsedilmiştir. Kafes yapısı ve kafes tabanlı kriptografide kullanılmakta olan zor problemler ve tezin ana konusunu oluşturan LWE problemi anlatılmıştır. Anahtar kapsülleme, açık anahtarlı şifreleme, Diffie – Hellman anahtar değişim protokolü ve ileri mükemmel gizlilik kavramından bahsedilmiştir.

Bölüm 3'de FrodoKEM protokolünün temel aldığı bazı protokoller ve FrodoKEM protokolü anlatılmıştır.

Bölüm 4'te FrodoKEM protokolü içerisindeki çarpım işlemleri temel alınarak oluşutulmuş olan matris vektör çarpım kütüphanesi ve bu matris vektör çarpım işlemleri yapılırken kullanılan yöntemler detaylandırılmıştır.

Bölüm 5'te Lizard, Emblem ve Lotus protokolleri detaylandırılmış ve oluşturulmuş olan matris vektör çarpım kütüphanesi bu protokollere uygulandığında elde edilen sonuçlardan bahsedilmiştir. Ek olarak bu protokollerin verimliliklerinin artırılması adına uygulamalarda yapılan değişikliklerden bahsedilmiştir. Son olarak ise oluşturulmak istenen tek sistem için yapılanlardan bahsedilmiştir.

Bölüm 6'da bölümde sonuç ve yapılması düşünülen gelecek çalışmalardan bahsedilmiştir.

2. MATEMATİKSEL ALT YAPI VE KRİPTOGRAFİK TERİMLER

Bu kısımda tez kapsamında kullanılan veya incelenen bazı terimler detaylandırılmıştır.

2.1. Matematiksel İfadeler ve Tanımlar

Bu bölümde kafes tabanlı sistemlerin matematiksel alt yapısını daha iyi ifade edebilmek için bazı tanımlar verilmiştir. Tez kapsamındaki genel gösterimler Çizelge 2.1. de gösterilmiştir.

Tanım 2.1.1: Herhangi bir A kümesinin çarpma veya toplamaya göre ("$\cdot$" işareti işlemi temsil ediyor);

- Kapalılık: $a, b \in A$ ve $a \cdot b \in A$,
- Birleşme: her $a, b, c \in A$ için $a \cdot (b \cdot c) = (a \cdot b) \cdot c$,
- Birim (etkisiz) eleman: her $a \in A$ için $a \cdot e = e \cdot a = a$ olacak şekilde bir $e \in A$ değeri olması,
- Her elemanın tersi: her $a \in A$ için öyle bir $a' \in A$ vardır ki $a \cdot a' = a' \cdot a = e$ olması.

A kümesi yukarıdaki özellikleri sağlıyorsa çarpmaya göre bir gruptur denir ve $< A, * >$ olarak gösterilir. A kümesi toplama işlemi için yukarıdaki özellikleri sağlarsa toplamaya göre bir gruptur denir ve $< A, + >$ şeklinde gösterilir.

Tanım 2.1.2: Herhangi bir A kümesi Tanım 2.1.1. deki özellikleri sağlayıp ek olarak;

- Değişme: Her $a, b \in A$ için $a \cdot b = b \cdot a$ özelliğini sağlıyorsa çarpmaya veya toplamaya göre değişmeli (Abel) gruptur.

Tanım 2.1.3: Herhangi bir A kümesi çarpma ve toplama işlemlerine göre Tanım 2.1.1 ve Tanım 2.1.2'deki özellikleri sağlayıp ek olarak;

- Sol ve sağdan dağılma: Her $a, b, c \in A$ için $a * (b + c) = (a * b) + (b * c)$ ve $(a + b) * c = (a * c) + (b * c)$ özelliğini sağlıyorsa A kümesi çarpma ve toplama işlemine göre bir halkadır denir ve $< A, *, + >$ olarak gösterilir.

Tanım 2.1.4: Herhangi bir $< A, *, + >$ toplama ve çarpmaya göre birim eleman ve değişme özelliği sağlıyorsa bu halka birimli ve değişmeli (Abel) bir halkadır denir.

Çizelge 2.1. Genel matematiksel gösterimler ve tanımları

Gösterim	Tanım
$< \mathbb{R}, + >$	Toplama işlemine göre bir grup
$< \mathbb{R}, * >$	Çarpma işlemine göre bir grup
$< \mathbb{R}, *, + >$	Toplama ve çarpma işlemine göre bir halka
$\mathbb{Z}_q^n$	Katsayıları $\mathbb{Z}_q$' nun elemanı olan n boyutlu bir vektör
$\mathbb{Z}_q^{m \times n}$	Katsayıları $\mathbb{Z}_q$' nun elemanı olan m×n boyutunda bir matris
$\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$	mod q'daki tam sayı elemanlar

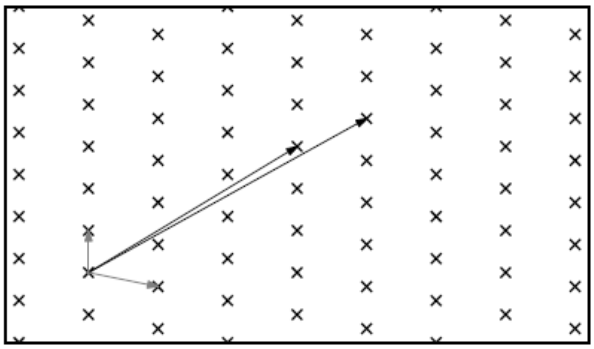
2.2. Kafes Yapısı

Bu bölümde kafes yapısının tanımı ve kafes yapısı ile ilgili bazı tanımlamalar yapılmıştır.

Tanım 2.2.1: Kafes periyodik bir yapıya sahip $\mathbb{R}^m$ olarak tanımlanan m boyutlu bir Öklid uzayındaki noktalar topluluğudur. Bir başka deyişle; n tane doğrusal olarak birbirinden bağımsız vektörün $< b_1, b_2, \dots, b_n \in \mathbb{R}^m >$ tam sayı kombinasyonlarının oluşturduğu kümedir ve aşağıdaki şekilde gösterilir (Regev, 2006).

$$\mathcal{L}\left(b_1, b_2, \dots, b_n\right) = \left\{ \sum_{i=1}^{n} x_i , b_i : x_i \in \mathbb{Z} \right\}$$

Buradaki n değeri kafesin rankı, m değeri ise kafesin boyutunu ifade etmektedir. Kafesi oluşturan $b_1, b_2, \dots, b_n$ vektörlerine kafesin tabanları (baz) denilmektedir. Aynı kafesi oluşturan birden fazla taban vektörü olabilir ve bu durum bir çok kriptografik sistemin alt yapısını oluşturan bir temeldir (Bernstein vd, 2009).



Şekil 2.1. İki boyutlu bir kafes yapısı ve bu kafesi oluşturan iki farklı vektör çifti (Regev, 2006)

Bir kafesi farklı şekilde vektörler oluşturabilir fakat bu taban vektörlerinin iyi ve kötü oldukları durumlar vardır. Bir kafesi oluşturan vektörler birbirleriyle ne kadar ortogonalse (dikse) o kadar iyi taban vektörleridir.

Tanım 2.2.2: Bir kafesin tabanı;

$$B = [b_1, \dots, b_n] \in \mathbb{R}^{n \times n}$$

şeklinde bir matris olarak ifade edilebilir. Matris içindeki her bir sütun, kafesin taban vektörlerini ifade etmektedir (Bernstein vd, 2009).

Tanım 2.2.3: $B \in \mathbb{Z}^{m \times n}$ şeklinde bir matris olsun, bu matristen üretilen bir kafes

$$\mathcal{L}(B) = \{B * x : x \in \mathbb{Z}^n\}$$

şeklinde tanımlanabilir. Burada $B * x$ çarpımı klasik matris vektör çarpımıdır (Bernstein vd, 2009; Peikert, 2016).

Tanım 2.2.4: Tam sayılardan oluşan bir matrisin determinantı ± 1 ise bu matrise ünimodüler matris denilmektedir. $B, B' \in \mathbb{Z}^{n \times n}$ iki matris olsun. Eğer $B' = B \cdot U$ eşitliğini sağlayabilecek bir unimodular U matrisi bulunabilior ise B ve B' matrisleri aynı kafesi oluşturmaktadır denir.

Tanım 2.2.5: $V \in \mathbb{R}^n$şeklinde bir vektör olsun. $\mathcal{L} \in \mathbb{R}^m$ kafesi içerisindeki sıfır olmayan en kısa vektör v'nin uzunluğu öklid uzaklığı kullanılarak bulunur ve

$$\lambda(\mathcal{L}) = \min_{v \in \mathcal{L}/\{0\}} \| v \|$$

şeklinde gösterilir.

2.3. Kafes Problemleri

Kafes tabanlı kriptografik sistemler, kafes yapısı üzerinde tanımlanmış olan zor problemler temel alınarak oluşturulan sistemlerdir. Kafes yapısı üzerinde en kısa vektör (SVP), en yakın vektör (CVP), hatalar ile öğrenme (LWE) gibi çözümlerinin zor olduğuna inanılan problemler bulunmaktadır. Oluşturulan kriptografik sistemlerin bir çoğunun güvenliği bu problemlerin zorluğuna dayanmaktadır (Hoffstein vd, 2008; Bernstein vd, 2009; Regev, 2010; Peikert, 2016).

Tanım 2.3.1: Herhangi bir B matrisi tabanlı $\mathcal{L} = \mathcal{L}(B)$ kafesi olsun. $v \in \mathcal{L}$ olup, sıfır olmayan en küçük vektörü bulma problemine en kısa vektör problemi (SVP) denilmektedir ve

$$\| v \| = \lambda(\mathcal{L})$$

olarak ifade edilmektedir (Peikert, 2016).

Tanım 2.3.2: Herhangi bir B matrisi tabanlı $\mathcal{L} = \mathcal{L}(B)$ kafesi ve $\mathcal{L}$ kafesinin içinde olmayan bir v vektörü verilmiş olsun. v vektörüne en yakın olan $v' \in \mathcal{L}$ vektörünü bulma problemine en yakın vektörü bulma problemi denilmektedir ve

$$\| v - v' \| = \lambda(\mathcal{L})$$

olarak ifade edilmektedir (Bernstein vd, 2009).

Tanım 2.3.3: ($SIS_{n,q,\beta,m}$ problemi (Peikert, 2016)). Büyük sonlu bir grup içerisindeki rastgele olarak üretilmiş elemanlardan, sıfıra ulaşan ve yeterli ölçüde kısa bir tam sayı kombinasyonu (vektör) bulma problemidir. Elimizde m, n, β tam sayıları olsun. m tane düzgün olarak rastgele üretilmiş vektör $v_i \in \mathbb{Z}_q^n$, $V \in \mathbb{Z}_q^{n \times m}$ matrisinin sütunlarını temsil etsin. Sıfır olmayan tam sayı bir vektör $a \in \mathbb{Z}^m$ ve $\| a \| \leq \beta$ olacak şekilde bir "a" vektörü bulmak amaçlanmaktadır.

$$f_V(a) := V \cdot a = \sum_i v_i \cdot a_i = 0 \in \mathbb{Z}_q^n$$

şeklinde gösterilir.

Tanım 2.3.4: (Bounded Distance Decoding Problemi (BDD_γ)). Tabanı B olan n-boyutlu bir $\mathcal{L}(B)$ kafesi ve uzaklığı $dist(t, \mathcal{L}) < d = \frac{\lambda_1(\mathcal{L})}{2\gamma(n)}$ şeklinde olan bir

$t \in \mathbb{R}^n$ hedef noktası olsun. $v \in \mathcal{L}$ olan $\| t - v \| < d$ şeklinde eşşiz bir v vektörünü bulma problemidir (Peikert, 2016).

2.3.1. Hatalar ile öğrenme problemi

LWE problemi 2005 yılında Oded Regev tarafından önerilmiş ve kriptografik protokoller için çok önemli bir temel olmuştur (Regev, 2005). Bu problem en kötü durumdaki kafes problemleri kadar zor olduğu varsayımı ile kriptografik yapıları güvence altına almakta olan bir problemdir. SIS probleminin şifreleme yapılmasını sağlayan bir türevidir (Peikert, 2016).

LWE problemi "secret" denilen $s \in \mathbb{Z}_q^n$ şeklinde bir vektörü, verilmiş olan rastgele doğrusal denklemlerden elde etme problemidir. Örnek olarak Şekil 2.2. deki denklemler verilmiş olsun. Burada her denklem eklenecek hata değeri kadar doğrudur (± 1 olsun) ve amaç s değerlerini elde etmektir. Bu denklem sistemi çözüldüğünde s = (0, 13, 9, 11) olmaktadır (Regev, 2010).

$$
\begin{aligned}
14s_1 + 15s_2 + 5s_3 + 2s_4 &\approx 8 \pmod{17} \\
13s_1 + 14s_2 + 14s_3 + 6s_4 &\approx 16 \pmod{17} \\
6s_1 + 10s_2 + 13s_3 + 1s_4 &\approx 3 \pmod{17} \\
10s_1 + 4s_2 + 12s_3 + 16s_4 &\approx 12 \pmod{17} \\
9s_1 + 5s_2 + 9s_3 + 6s_4 &\approx 9 \pmod{17} \\
3s_1 + 6s_2 + 4s_3 + 5s_4 &\approx 16 \pmod{17} \\
&\vdots \\
6s_1 + 7s_2 + 16s_3 + 2s_4 &\approx 3 \pmod{17}
\end{aligned}
$$

Şekil 2.2. LWE problemi için örnek bir denklem sistemi (Regev, 2010)

Eğer denklemlere eklenen hata değeri olmasaydı, buradan s değerleri, Gauss elemesi yöntemi kullanılarak polinom zamanda elde edilebilirdi. Hata değeri eklenmesi burada bu denklem sistemini çözülmesi zor bir matematiksel probleme dönüştürmektedir (Regev, 2006; Regev, 2010).

LWE problemi Regev tarafından parametreleri ile beraber şu şekilde tanımlanmıştır: sabit bir boyut parametresi $n \geq 1$, mod değeri $q \geq 2$ ve hata olasılık dağılımı $\chi \in \mathbb{Z}_q$ olsun. $A_{s,\chi} \in \mathbb{Z}_q^n \times \mathbb{Z}_q^n$; rastgele olarak seçilen bir $a \in \mathbb{Z}_q^n$

vektörü seçilerek elde edilen olasılık dağılımı, χ değerine göre elde edilen hata değeri $e \in \mathbb{Z}_q^n$, çıktı olarak $(a, < a, s > + e)$ ve bütün matematiksel indirgemeler mod q altında yapılıyor olsun. LWE problemi mod q üzerinde hata dağılımı χ olacak şekilde, herhangi bir $s \in \mathbb{Z}_q^n$ değeri için, rastgele ve birbirinden bağımsız $A_{s,\chi}$ dağılımı içerisindeki örnekler üzerinden s değerini elde etmeye çalışmaktır (Regev, 2005; Regev, 2010). Ek olarak özel bir durum olarak; mod değeri $q = 2$ seçilirse bu problem Learning Parity With Noise (LPN) problemi olarak karşımıza çıkmaktadır (Pietrzak, 2012). Ayrıca LWE problemi $A \in \mathbb{Z}_q^{n \times m}$ şeklinde bir matris üzerinden üretilen bir kafes için ortalama durumdaki bir Bounded Distance Decoding (BDD) problemi olarak da görülebilir (Regev, 2010; Peikert 2016). LWE problemin "Arama" ve "Karar Verme" şeklinde iki çeşidi bulunmaktadır.

Tanım 2.3.4: $Arama - LWE_{n,q,\chi,m}$ problemi; elimizde $A_{s,\chi}$ dağılımından elde edilen m tane $a_i, b_i \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ ve rastgele olarak üretilmiş $s \in \mathbb{Z}_q^n$ değerleri (s her örnek için sabit) üzerinden s değerini bulma problemidir (Peikert, 2016).

Tanım 2.3.5: $Karar\ Verme - LWE_{n,q,\chi,m}$ problemi; problemi; elimizde $A_{s,\chi}$ dağılımından elde edilen m tane $a_i \in \mathbb{Z}_q^n$ ve rastgele olarak üretilmiş $s\ ve\ b \in \mathbb{Z}_q^n$ değerleri olsun. Üretilen LWE örnekleri ile rastgele olan değerleri birbirinden ayırma problemidir (Peikert, 2016).

LWE problemlerinin bu versiyonlarını matrisler üzerinde tanımlayacak olursak; $a_i \in \mathbb{Z}_q^n$ vektörleri $A \in \mathbb{Z}_q^{n \times m}$ matrisinin sütunlarını ifade etsin, $e_i \in \mathbb{Z}$ ve $b_i \in \mathbb{Z}_q$ değerleri sırasıyla $e \in \mathbb{Z}^m$ ve $b \in \mathbb{Z}_q^m$ vektörlerindeki değerleri temsil etsin. $A, b = A\,s + e\ mod\ q$ değerlerinden s yi elde etme (Arama-LWE) veya rastgele olan (A, b) değerleri ile LWE örneklerini birbirinden ayırt etme (Karar Verme-LWE) olarak tanımlanmaktadır (Peikert, 2016).

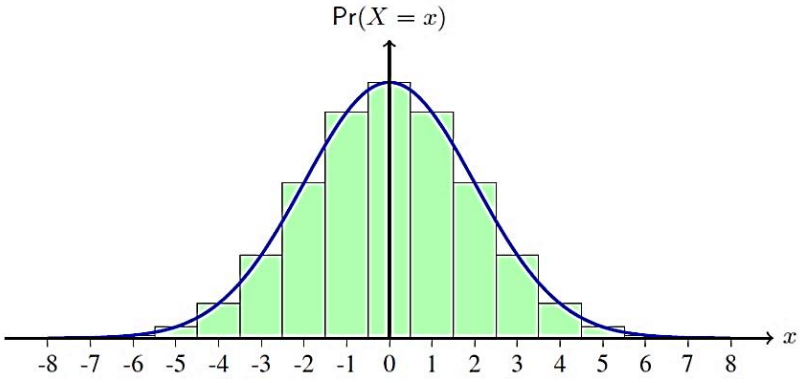
LWE probleminin zorluğu. Bu problemin zor bir problem olarak tanımlanmasının sebeplerinden ilki; LWE problemini polinom zamanda çözebilen herhangi bir algoritma (kuantum algoritmalar dahil) bulunmamasıdır. Diğer bir sebep çözülmesi zor bir problem olarak görülen LPN problemine olan benzerliğidir. Bir diğer ve en önemli sebep; yeterince büyük bir q değeriyle arama-LWE probleminin en kısa vektör problemi ve türevleri kadar zor bir problem olmasıdır (Peikert, 2009). q değeri yeterince düzgün seçilirse (örneğin iki asal sayını çarpımı şeklinde) karar verme-LWE problemi en az arama-LWE kadar zor bir problem olmaktadır (Regev, 2005;

Peikert, 2009; Lindner ve Peikert, 2011). Ek olarak S değerini rastgele olarak seçmek veya ayrık gauss dağılımı ile üretmek problemin zorluğu üzerinde bir değişiklik yapmamaktadır.

2.4. Ayrık Gauss Dağılımı

Bu kısımda tez kapsamında incelenen ve kriptografik protokollerde kullanılmakta olan örnekleme yöntemlerinden kısaca bahsedilecektir.

Tanım 2.4.1: (Ayrık Gauss Dağılımı). LWE problemini temel alan kriptosistemlerde, gizli anahtarı saklı tutmak için ayrık Gauss hatası eklenmesi gerekmektedir.



Şekil 2.3. Ayrık Gauss dağılımı. Yeşil çubuklar olasılık kütlesini (probabilty mass), mavi çizgi ise karşılığındaki sürekli olasılık yoğunluk (continuous probablity density) fonksiyonunu ifade etmektedir (Saarinen, 2015)

Ayrık Gauss dağılımı (ayrık normal dağılım) $D_{\mathbb{Z},\sigma}$ tam sayılar üzerinde sapma parametresi ile tanımlanmaktadır. $x \in \mathbb{Z}$ şeklinde bir elemanın olasılığı (normal dağılım)

$$\rho_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}}$$

ile orantılıdır (Saarinen, 2015).

$$S_\sigma(\mathbb{Z}) = \sum_{k=-\infty}^{\infty} \rho_\sigma(k) \approx \sigma\sqrt{2\pi}$$

şeklinde tek yönlü bir fonksiyon olsun. $\alpha \in \mathbb{Z}$ şeklinde bir değerin $D_{\mathbb{Z},\sigma}$ dağılımından örneklenmesi

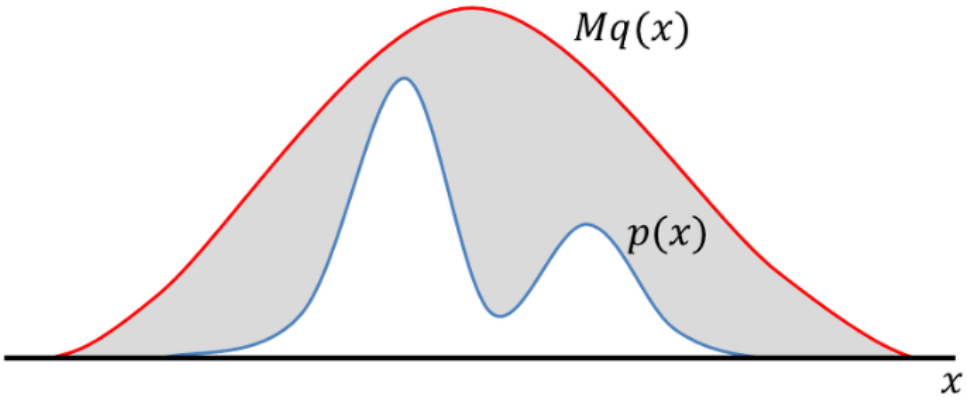
$$\rho_\sigma(x) = \Pr(X = x) = \frac{\rho_\sigma(x)}{S_\sigma(\mathbb{Z})} \approx \frac{e^{-\frac{x^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

şeklinde hesaplanır. Ayrık Gauss dağılımı için bazı parametrelere ihtiyaç duyulmaktadır. Bu parametreler ortalama (μ – ortalama), standart sapma (σ – standard deviation), hassasiyet (λ - precision) ve kuyruk kesimi (τ – tail cut) parametreleridir ve kullanımları şu şekildedir (Howe vd, 2016, Howe ve O'Neill, 2017);

- Ortalama: Normal dağılımın orta noktasıdır. Kafes tabanlı kriptografide ortalama genellikle 0 olarak kullanılmaktadır.
- Standart sapma: Dağılımın şeklini kontrol eder. LWE veya SIS problemlerinde indirgeme yapılan değere bağlıdır.
- Hassasiyet: Güvenlik seviyesi ג olmak üzere, teorideki mükemmel dağılım ile uygulama arasındaki istatistiksel mesafeyi gerekli olan güvenlik seviyesi $2^{-\lambda}$ değerinden büyük olmayacak şekilde ayarlamak için kullanılır.
- Kuyruk Kesimi: Belirli bir güvenlik seviyesi için x eksenindeki dışlamayı yönetir.

Tanım 2.4.2: (Reddetme Örneklemesi (Rejection Sampling) (Acceptance-Rejection Method) (Howe vd, 2016)). Kafes tabanlı kriptosistemlerde ayrık Gauss dağılımı ile hata örneklemesi yapmak için önerilmiş olan ilk yöntemdir (Gentry vd, 2008). Bu yöntem ile bir olasılık dağılımı (q) verildiğinde isteğe bağlı bir hedef dağılım (p) üretilebilmektedir. g dağılımından bir örnek $p(x)/(M \cdot q(x))$ ($M \in \mathbb{R}^+$) olasılığı ile $p(x)/q(x)$ sınırı altında kabul edilebilir. Ayrık Gauss dağılımından örnek üretmek için reddetme örneklemesi kullanılırken, düzgün bir rastgele değer x, rastgele bir değer $\mu \in [0,1)$ seçilir ve $\mu < \rho_\sigma(x)$ olup olmadığına bakılır. Eğer rastgele μ değeri Gauss dağılımı eğrisinin altında ise örnek kabul (accept), eğrinin üstünde ise örnek red (reject) edilir. Bu yöntem kullanılırken örnek kabul edilene kadar ortalama olarak $2\tau/\sqrt{2\pi}$ deneme yapılması gerekmektedir. Yöntem, kabul edilecek değere ulaşana kadar çok fazla reddetme işlemi gerektirdiğinden çok

maliyetlidir. Düşük verimli bir yöntem olduğundan uygulamalarda çok fazla tercih edilmeyen bir yöntemdir.



Şekil 2.4. Reddetme örneklemesi (Rejection Sampling, 2018)

Tanım 2.4.3: (Kümülatif Dağılım Örneklemesi (Cumulative Distribution (CDT) Sampling) (Howe vd, 2016)). İçerisinde önceden hesaplanmış CDF değerleri bulunan büyük CDT'ler ile ayrık Gauss dağılımı örneklemesi yapılması fikri ilk olarak Peikert (Peikert, 2010) tarafından ortaya atılmış, Ducas ve arkadaşları tarafından BLISS (Ducas vd, 2013) isimli imzalama şemasında uygulanmıştır. Bu yöntemde normal dağılımdaki simetri, $\mathbb{Z}^+$ dan örnek almak için gereken tablo boyutunun yarısından tasarruf etmek için kullanılır. İlk değeri 0 ve son değeri 1 olacak şekilde toplamda $N = \sigma \times \tau$ tane örnek ile

$$0 = S[0] < S[1] < \cdots < S[N-3] = 1$$

şeklinde bir tablo oluşturulur. Devamında CDF değerleri ($S[\cdot]$) hesaplanır. Bir örnek $r \in [0,1)$ λ bitlik bir hassasiyetle düzgün şekilde üretilir. Üretilecek örnek, istenen aralıkta $S[x] \leq r < S[x+1]$ şeklinde ve $\rho[x] = S[x+1] - S[x]$ olasılığı ile üretilir. İlk tablodaki değerler, son aşamada oluşan değerlerden daha tahmin edilebilirdir. Oluşturulan tablo sıralı bir tablo olduğundan hedef değerin yerini bulmak için ikili arama algoritması kullanılır. Başlangıçta tablonun tamamını içeren arama alanı, ikili arama algoritması çalıştıkça her seferinde yarıya düşerek azalır. Bu şekilde CDT üzerinden ayrık Gauss dağılımı örneklemesi yapılmış olur.

$Girdi: min, cur\ ve\ jmp$ üç $tane\ tam\ sayı,$ $Ayrık\ Gauss\ CDT$ örnekleri; $Tablo\ boyutu\ (N) = \sigma \times \tau,$ $Tablo: 0 = S[0] < S[1] < \cdots < S[N-3] = 1$
1: $Tek\ bir\ b\ \in \{0,1\}\ biti$ örneklenir 2: $Rastgele\ r \in \{0, \ldots, (2^{\lambda} - 1)\}\ değeri\ oluşturulur.$ 3: $min \leftarrow 0, cur \leftarrow N/2, jmp \leftarrow cur$ 4: $while\ (jmp > 0)\ do$ 5: $cur \leftarrow min + jmp$ 6: $Eğer\ (r \geq S[cur]$ 7: $min \leftarrow cur$ 8: $Aksi\ halde$ 9: $jmp \leftarrow jmp \gg 1$ 10: $Çıktı\ x = (-1)^b min$

Şekil 2.5. CDT örneklemesi sözde kodu (Howe vd, 2016)

2.5. Kriptografik Altyapı

Bu kısımda tez kapsamında incelenmiş olan kriptografik sistemler ile ilgili tanımlar ve Diffie – Hellman anahtar değişim protokolünden bahsedilmiştir.

2.5.1. Açık anahtarlı şifreleme

Bir açık anahtarlı şifreleme şeması şifre metin uzayı C ve mesaj uzayı M ile birlikte, Tanımlama (Setup), Anahtar Üretimi (Key Generation), Şifreleme (Encryption) ve Şifre Çözme (Decryption) olmak üzere dört tane algoritmadan oluşmaktadır. Bu işlemler sırasıyla şu şekildedir (Peikert, 2014);

- Tanımlama (Setup); Çıktı olarak açık parametreleri (public parameters – pp) verir.
- Anahtar Üretimi (Key Generation); Gen(pp) olarak gösterilir. Çıktı olarak açık şifreleme anahtarı (public key – pk) ve gizli olan şifre çözme anahtarını (secret key – sk) verir.
- Şifreleme (Encryption); Enc(pp, pk, μ) olarak gösterilir. Açık anahtarı ve $\mu \in M$ şeklinde iletilecek bir mesajı girdi olarak alır, çıktı olarak $c \in C$ şeklinde bir şifre metin verir.
- Şifre Çözme (Decryption); Dec(sk, c) olarak gösterilir. Girdi olarak gizli anahtarı ve şifre metni alır. Çıktı olarak $\mu \in M \cup \{\perp\}$ şeklinde bir çıktı verir. μ burada düz metin veya şifre çözme hatası olan $\perp$ şeklindedir.

2.5.2. Diffie-Hellman anahtar değişim protokolü

Bir anahtar değişim protokolünün amacı, tarafların yalnızca kendilerinin bildiği bir oturum anahtarı oluşturmaktır (Diffie vd, 1992). Bu protokol ayrık logaritma probleminin zorluğuna dayanmaktadır.

Tanım 2.5.1: (Ayrık Logaritma Problemi). $x \in \mathbb{Z}_p$ sayısı, p asal sayı ve $a, b \in \mathbb{Z}_p^*$ olsun. Yeterince büyük bir p asal sayısı seçildiği zaman

$$b \equiv a^x \bmod p$$

denkleminde b değeri biliniyor olsa bile, x değerinin bulunması zor bir problem olarak sınıflandırılmaktadır. Bu x değerinin bulunması problemine ayrık logaritma problemi denilmektedir.

DH anahtar değişimi, iki taraf arasında, tarafların yanlızca kendi gizli anahtarlarını bildikleri bir durumda ortak bir oturum anahtarı oluşturma protokolüdür (Diffie ve Hellman, 1976). Simetrik şifreleme sistemlerinin çoğunda anahtar değişim işlemi bu protokol ile yapılmaktadır. Protokolün genel yapısı Şekil 2.6'da verilmiştir.

Ayrık logaritma probleminin Shor algoritmasından sonra kuantum bilgisayarlar ile polinom zamanda çözüleceği öngörüldüğünden, kuantum sonrası için polinom zamanda çözülemeyen yeni problemlere dayalı anahtar değişim şemalarına ihtiyaç bulunmaktadır.

A		**B**
p, g	Adım 1: Asal ve üretici seçimi	p, g
a	Adım 2: Gizli anahtarların üretilmesi	b
A = $g^a mod\ p$	Adım 3: Açık anahtarların üretilmesi	B = $g^b mod\ p$
k = $B^a mod\ p$ k = $(g^b)^a mod\ p$ k = $g^{ab} mod\ p$	Adım 4: Oturum anahtarı üretilmesi	k = $A^b mod\ p$ k = $(g^a)^b mod\ p$ k = $g^{ab} mod\ p$

Şekil 2.6. DH anahtar değişim protokolü genel yapısı

2.5.3. İleri mükemmel gizlilik

Eğer bir saldırgan, kriptografik bir sistemde kullanılmakta olan uzun süreli gizli anahtarı bir şekilde elde edebilirse bu durum taraflar arasındaki iletişimin güvenliği açısından büyük problemlere yol açabilir. Bu durumun engellenmesi adına ileri mükemmel gizlilik kavramı ortaya çıkmıştır. Bu kavram basitçe; şifreli olarak gizli kabul edilen bir şeyin ilerleyen zamanlarda da şifreli kalması ve gelecekte bulunmasının kolay olmadığı bir durumu ifade etmektedir (Boneh ve Shoup, 2015). PFS anahtar değişiminde kullanılan oturum anahtarının herhangi bir şekilde kaybolması veya açığa çıkması durumunda güvenli iletişimi korumak için, bir ara geçici şifreleme kullanarak veya kullanılan oturum anahtarının her seferinde yenilenmesi ile sağlanmaktadır. Saldırgan o anda kullanılan oturum anahtarını elde etse bile taraflar arasındaki iletişimin tamamı hakkında bilgi edinemez. Bu şekilde taraflar arasındaki iletişimin güvenliğini sağlamak amaçlanır.

2.5.4. Anahtar kapsülleme/paketleme protokolleri

Anahtar kapsülleme protokolleri açık anahtarlı şifreleme yöntemlerini kullanarak, karşı tarafın açık anahtarı ile geçici bir gizli anahtarı (oturum anahtarı) iletmek için kullanılan protokollerdir (Peikert, 2014). Basit olarak anahtar değişim protokolleridir. Anahtar değişim protokolleri; taraflar arasında yapılan karşılıklı etkileşimli protokollerdir. Anahtar değişimi protokollerinin amacı; taraflar arasında kullanılabilecek, rastgele ayırt edilemez bir oturum anahtarı oluşturmaktır (Datta vd, 2006). Anahtar değişimi protokollerinde, işlemi yapacak iki tarafta oturum anahtarını oluşturmaya katkıda bulunur. Anahtar kapsülleme protokollerinde ise tek taraflı bir işlem yapılmaktadır. Kısaca; bir taraf oturum anahtarını üretir, şifreler ve karşı tarafa gönderir. Bu yapılarıyla anahtar değişim protokollerinden farklı olmaktadırlar.

Bir anahtar kapsülleme protokolü şifre metin uzayı C ve anahtar uzayı K ile birlikte Anahtar Üretimi, Kapsülleme ve Kapsülü Çözme şeklinde üç tane algoritmadan oluşmaktadır ve $\Pi = (\mathrm{K}, \mathrm{E}, \mathrm{D})$ şeklinde gösterilmektedir (Peikert, 2014; Coretti vd, 2013). Bu algoritmalardan önce uygulanan Tanımlama (Setup) adında herkese açık olarak kullanılan parametrelerin (public parameters) (pp) eklendiği bir kısımda bulunmaktadır (Peikert, 2014). Tanımlama işleminden sonra sırasıyla şu işlemler yapılır (Peikert, 2014; Coretti vd, 2013);

- Anahtar Üretimi (Key Generation) : Gen(pp) şeklinde gösterilir. Bu kısımda açık parametreler girdi olarak alınır, açık (pk) ve gizli (sk) anahtar ikilisi üretilir.
- Kapsülleme (Encapsulation): Encaps(pp, pk) şeklinde gösterilir. Açık parametreler ile önceki aşamada üretilmiş olan açık anahtar girdi olarak alınır, çıktı olarak $c \in C$ şeklinde bir şifre metni ve $k \in K$ olan bir oturum anahtarı oluşturulur.
- Kapsülü Çözme (Decapsulation): Decaps(sk, c) şeklinde gösterilir. Gizli anahtar ve şifre metni girdi olarak alır. Çıktı olarak $k \in K \cup \perp$ şeklinde bir k değeri verir. k değeri burada oturum anahtarı veya hata sembolü olan $\perp$ değeri şeklindedir.

3. FrodoKEM PROTOKOLÜ VE FrodoKEM PROTOKOLÜNÜN TEMEL ALDIĞI PROTOKOLLER

2005 yılında, Oded Regev LWE problemini, bu problemin zorluğunu kullanan bir şifreleme sistemi önermiştir ve LWE probleminin en kötü durumdaki kafes problemleri kadar zor olduğunu göstermiştir (Regev, 2005). 2011 yılında LWE problemine farklı bir bakış açısı olarak, Lindner ve Peikert tarafından yapılan bir çalışmada; dikdörtgen şeklinde matrisler yerine kare matris kullanmak gibi birçok değişiklik yapılarak daha verimli bir LWE tabanlı şifreleme sistemi önerilmiştir (Lindner ve Peikert, 2011). Bu çalışmalardan sonra 2016 yılında Regev ve Lindner – Peikert şifreleme sistemlerini temel alan bir sistem olan Frodo (Bos vd, 2016) anahtar değişim protokolü önerilmiştir. Devamında NIST'in çağrısı sonrasında Frodo protokolünü temel alan, LWE problemini kullanan ve bu çalışmanın ana kapsamını oluşturan FrodoKEM (Naehrig vd, 2017) anahtar kapsülleme protokolü önerilmiştir. Bu kısımda FrodoKEM protokolünün temelini oluşturan Regev, Linder-Peikert, Frodo protokolleri ve tezin ana konusunu oluşturan FrodoKEM protokolü anlatılacaktır.

3.1. Regev'in Açık Anahtarlı Şifreleme Sistemi

Regev tarafından oluşturulmuş olan kriptosistem içerisinde; n güvenlik parametresini (LWE boyutu), q mod alınan değeri, χ değeri $\mathbb{Z}_q$ üzerindeki hata dağılımını ve m değeri ise LWE örneği sayısını ifade etmektedir.

Gizli anahtar $s \in \mathbb{Z}_q^n$ rastgele seçilen bir vektör ve açık anahtar $m \approx (n+1)\log q$ tane, $(\bar{a}_i, b_i = < s, \bar{a}_i > + e_i) \in \mathbb{Z}_q^{n+1}$ şeklinde $A_{s,\chi}$ dağılımından üretilmiş olan vektörlerdir (Regev, 2009).

$$A = \begin{bmatrix} \bar{A} \\ b^t \end{bmatrix} \in \mathbb{Z}^{(n+1)\times m}$$

şeklinde ifade edilebilir ve burada $b^t = s^t\bar{A} + e^t \, mod \, q$ şeklindedir. Açık anahtar ve gizli anahtar arasında

$$(-s, 1)^t \cdot A = e^t \approx 0 \, (mod \, q)$$

şeklinde bir ilişki bulunmaktadır (Peikert, 2009). Sistemde tek bir bitin şifrelenmesi ve şifresini çözülmesi işlemleri yapılmaktadır. $\mu \in \mathbb{Z}_2 = \{0,1\}$ şeklinde bir biti A

açık anahtarı ile şifrelemek için; LWE örnekleri arasından rastgele bir ϑ alt kümesi seçilir ve mesaj biti bu alt kümenin son kısmına uygun şekilde yerleştirilir. Şifrelenecek bit 0 ise $(\sum_{i\in\vartheta} a_i, \sum_{i\in\vartheta} b_i)$, şifrelenecek bit 1 ise $(\sum_{i\in\vartheta} a_i, \left\lfloor\frac{q}{2}\right\rfloor + \sum_{i\in\vartheta} b_i)$ şeklinde bir alt küme seçilir. (Regev, 2005)

$$c = a \cdot b + \left(0, \mu \cdot \left\lfloor\frac{q}{2}\right\rfloor\right) \in \mathbb{Z}_q^{n+1}$$

şeklinde gösterilebilir. Gizli anahtar s ile şifrelenmiş bitin şifre çözme işlemi;

$$(-s, 1)^t \cdot c = (-s, 1)^t \cdot a \cdot b + \mu \cdot \left\lfloor\frac{q}{2}\right\rfloor$$

$$= e^t \cdot x + \mu \cdot \left\lfloor\frac{q}{2}\right\rfloor$$

$$\approx \mu \cdot \left\lfloor\frac{q}{2}\right\rfloor \ (mod\ q)$$

şeklinde yapılmaktadır. Şifre çözme işleminin sonucu; $b - \langle a, s\rangle$ işleminin sonucu 0 dan çok $\left\lfloor\frac{q}{2}\right\rfloor$ değerine yakınsa 0, değilse 1 olmaktadır (Regev, 2005; Peikert, 2009).

3.2. Lindner-Peikert Açık Anahtarlı Şifreleme Sistemi

Bu kriptosistem içerisinde; tam sayı şeklinde, $q \geq 2$ mod alınan değeri ve $n_1, n_2 \geq 1$ boyut parametrelerini, anahtar üretimi için σ_k ve şifreleme için σ_e, mesaj uzayı Σ ve mesaj boyutu $\ell \geq 1$ dir. Ek olarak hata terimlerinin etkisini ortadan kaldıran basit bir kodlayıcı ve kod çözücü sistemde bulunmaktadır.

Bu sistemde en küçük boyutta açık anahtarlar elde etmek için, herkese açık olan düzgün ve rastgele bir şekilde üretilmesi gereken $A \in \mathbb{Z}_q^{n_1 \times n_2}$ matrisini, sistemde bulunan güvenli bir dış kaynağa ürettirmektedir. Eğer sistemde güvenilir bir dış kaynak yoksa A matrisi anahtar üretimi aşamasında kullanılabilir ve açık anahtara dâhil edilebilir (Peikert, 2009).

Anahtar Üretimi $(\mathrm{Gen}(A, 1^\ell))$: $E \leftarrow D_{\mathbb{Z}, s_k}^{n_1, \ell}$ ve $S \leftarrow D_{\mathbb{Z}, s_k}^{n_2, \ell}$ ve $B = E - AS$ şeklinde olsun. Açık anahtar burada B ve gizli anahtar S değerleridir. Açık anahtar ve gizli anahtar arasındaki ilişki

$$[A \quad B] \cdot \begin{bmatrix} S \\ I \end{bmatrix} = E\ mod\ q$$

şeklindedir (Peikert, 2009).

Şifreleme $(Enc(A, B, m \in \Sigma^{\ell}))$: $e = (e_1, e_2, e_3) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \times \mathbb{Z}^{\ell}$ şeklinde her biri birbirinden bağımsız olmak üzere $D_{\mathbb{Z},s_e}$ dağılımı üzerinden elde edilir. Ek olarak $\bar{m} = kodlayıcı(m) \in \mathbb{Z}_q^{\ell}$ şeklinde bir mesaj seçilir. Şifre metnin hesaplanması

$$c^t = [c_1^t \quad c_2^t] = [e_1^t \quad e_2^t \quad e_2^t + \bar{m}^t] \cdot \begin{bmatrix} A & B \\ I & \\ & I \end{bmatrix} \in \mathbb{Z}_q^{1\times(n_2+\ell)}$$

şeklinde yapılmaktadır.

Şifre Çözme ($Dec(c^t = [c_1^t, c_2^t], S)$):

$$[c_1^t \quad c_2^t] \cdot \begin{bmatrix} S \\ I \end{bmatrix} = (e^t + [0 \quad 0 \quad \bar{m}^t]) \cdot \begin{bmatrix} E \\ S \\ I \end{bmatrix} = e^t \cdot \begin{bmatrix} E \\ S \\ I \end{bmatrix} + \bar{m}^t$$

işleminin sonucunun kod çözücüye (decoder) verilmesi ile yapılmaktadır.

3.3. Frodo Anahtar Değişim Protokolü

Frodo protokolü standart kafesler üzerindeki LWE probleminin anahtar değişim protokolleri üzerindeki ilk uygulaması olarak öne çıkmakta olan bir protokoldür (Nejatollahi vd, 2019). $B, s, \bar{m}, \bar{n}, n, q \in \mathbb{Z}$ ve $\sigma > 0$ şeklinde bir gerçel sayıyı parametre olarak kullanır (Yuan vd, 2018). $A \in \mathbb{Z}_q^{n \times n}$ matrisi, $seed_A$ değeri kullanılarak sözde rastgele sayı üreteci fonksiyonu $Gen()$ aracılığıyla üretilir.

Çizelge 3.1. Frodo anahtar değişim protokolü genel yapısı (Bos vd, 2016)

X		Y
$seed_A \xleftarrow{\$} U(\{0,1\}^s)$ $A \leftarrow Gen(seed_A)$ $S, E \xleftarrow{\$} \chi(\mathbb{Z}_q^{n \times \bar{n}})$ $B \leftarrow AS + E$		
	$seed_A,\ B \in \{0,1\}^s \times \mathbb{Z}_q^{n \times \bar{n}} \longrightarrow$	
		$A \leftarrow Gen(seed_A)$ $S', E' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times n})$ $B' \leftarrow S'A + E'$ $E'' \xleftarrow{\$} \chi(\mathbb{Z}_q^{\bar{m} \times \bar{n}})$ $V \leftarrow S'B + E''$ $C \leftarrow \langle V \rangle_{2^B}$
	$B',\ C \in \mathbb{Z}_q^{\bar{m} \times n} \times \mathbb{Z}_2^{\bar{m} \times \bar{n}} \longleftarrow$	
$K \leftarrow rec(B'S, C)$		$K \leftarrow [V]_{2^B}$

Protokoldeki X ve Y tarafları LWE örneklerini hesaplamak için aynı ve büyük boyutta bir $A \in \mathbb{Z}_q^{n \times n}$ matrisini üretirler. X tarafından $\bar{n}$ tane ve Y tarafından $\bar{m}$ tane LWE örneği $\mathbb{Z}_q^{\bar{m} \times \bar{n}}$ içerisinde gizli bir matris oluşturmak için birleştirilir ve oturum anahtarını oluşturmak için sisteme her örnek girişinde B tane bit girişten çıkartılır. Bundan dolayı $\bar{m}$ ve $\bar{n}$ boyutları oturum anahtarı için gerekli hedef güvenlik seviyesini sağlayacak bit sayısına sahip olacak şekilde seçilmektedir (Bos vd, 2016). Frodo protokolünde 128-bit kuantum sonrası güvenliği sağlamak için kullanılan matrislerin boyutları $\bar{m} \cdot \bar{n} \cdot B \geq 256$ olacak şekilde seçilmiştir. Bu şekilde 256-bit düzgün oluşturulmuş bir oturum anahtarı elde edilir ve Grover'ın ayrıntılı anahtar arama algoritması üretilen anahtarı 2^{128} aramada ancak elde edebilmektedir. Ek

olarak açık anahtarın ve şifre metnin oluşturulması aşamasında matris vektör çarpım işlemleri bulunmaktadır. Frodo protokolü hakkında bahsedilmesi gereken diğer bir konu ise protokolde kullanılan uzlaşma mekanizmasıdır. Bu uzlaşma mekanizması, Peikert'ın 2014 yılında yaptığı bir çalışmadaki mekanizmanın genelleştirilmiş bir türüdür (Peikert, 2014). Çizelge 3.1. de görülen sistemde X ve Y tarafları sırasıyla $V \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ ve $B'S \in \mathbb{Z}_q^{\bar{m} \times \bar{n}}$ matrislerini üretir. Y tarafı X tarafına $C = \langle V \rangle_{2^B}$ işlemini yaparak sonucunu gönderir. $\bar{B} = (\log_2 q) - B$ değeri $M \in \mathbb{Z}_q^{x \times y}$ matrisi için tanımlanmış olsun ve *yuvarlama* fonksiyonu $[M]_{2^B}$ ve *çapraz - yuvarlama* fonksiyonu $\langle M \rangle_{2^B}$ şeklinde tanımlanmıştır. Bu fonksiyonlar;

$$yuvarlama : [M]_{2^B} = \left[2^{-\bar{B}} * M\right] mod\ 2^B$$

$$çapraz\text{ - }yuvarlama: \langle M \rangle_{2^B} = \left\lfloor 2^{1-\bar{B}} * M \right\rfloor mod\ 2$$

şeklinde gösterilmektedir. $\lfloor \cdot \rceil$ işlemi burada orta değere yuvarlama işlemini temsil etmektedir. Uzlaşma fonksiyonu olan $rec()$ Peikert'ın 2014 yılında yaptığı çalışmadaki fonksiyondur (Peikert, 2014). Çıktı v, w $\in \mathbb{Z}_q$ içerisindeki en yakın elemandır ve $\langle v \rangle_{2^B} = 0$ veya $\langle v \rangle_{2^B} = 1$ şeklindedir (Yuan vd, 2018). Taraflar bu işlemleri ve fonksiyonları kullanarak K oturum anahtarı üzerinde anlaşma işlemini yapmaktadırlar.

3.4. FrodoKEM Anahtar Kapsülleme Protokolü

FrodoKEM protokolü; açık anahtarlı şifreleme tekniklerini kullanmakta olup Regev ve Lindner-Peikert şifreleme sistemleri ile önceki versiyonu sayılabilecek Frodo protokolünü temel alan bir anahtar kapsülleme protokolüdür. 2017 yılında NIST'in kuantum sonrası güvenli sistemler için yaptığı çağrı sonrası önerilmiş olan bir protokoldür. NIST projesinde, standart LWE problemini kullanan anahtar kapsülleme protokolleri arasında ikinci tura geçmiş olan tek protokol olarak öne çıkmaktadır.

FrodoKEM protokolünün NIST projesinde önerilen diğer anahtar kapsülleme protokollerine göre öne çıkan özellikleri bulunmaktadır. Bu özelliklerden bazıları şunlardır;

- **Modüler İndirgeme:** FrodoKEM protokolünün parametre setlerinde $q \leq 2^{16}$ şeklinde ikinin kuvveti olan bir tam sayı, modüler indirgeme işlemleri için kullanılmaktadır. Bu durum tek hassasiyetli aritmetik işlemlerin kullanılmasını sağlamaktadır ve indirgeme işlemi bit – maskeleme ile 16 bitlik değişkenler kullanıldığında neredeyse zaman harcamayan bir işlem olmaktadır (Naehrig vd, 2017).
- **Matris Vektör İşlemleri:** FrodoKEM protokolü içinde basit matris vektör çarpım işlemleri bulundurmaktadır. Matris çarpımı yerine polinom çarpımı kullanan sistemlere göre daha uzun bir çalışma zamanı gerektirse de uygulaması bu sistemlere oranla daha kolaydır (Naehrig vd, 2017). Ayrıca diğer anahtar kapsülleme protokollerine bakıldığında FrodoKEM içerisindeki matris vektör işlemleri uygulamalar üzerinde çok daha verimli çalışmakta olan bir yapıya sahiptir.
- **Dinamik Olarak Oluşturulan Matrisler:** Açık anahtarların boyutunu düşürmek için ve şifreleme işlemini hızlandırmak amacıyla oluşturulan her anahtar için yeni bir $A \in \mathbb{Z}_q^{n \times n}$ matrisi oluşturulmaktadır. Sözde rastgele sayı üretme, tüm matrisin hızlı bir şekilde oluşturulmasına ya da kısıtlı belleği olan sistemlerde matrisi satır satır oluşturacak şekilde tanımlanmış bir işlemdir (Naehrig vd, 2017).

Bu gibi özelliklerden dolayı tez kapsamında FrodoKEM protokolü ve protokolün uygulamasında yapılan işlemler, iyileştirmeler ve teknikler incelenmiştir.

3.4.1. FrodoKEM protokolünde kullanılan parametreler

FrodoKEM anahtar kapsülleme protokolü mod q altındaki $\mathbb{Z}_q$ tam sayı halkası üzerinde çalışmaktadır. Kullanılan parametreler Frodo protokolü ile çoğu yerde benzerlik göstersede uygulama içerisinde değişiklik gösteren durumlar bulunmaktadır. FrodoKEM protokolünün uygulamasında üç farklı güvenlik seviyesi için üç farklı parametre seti tanımlanmıştır. Bu parametreler; AES-128 güvenlik seviyesi için FrodoKEM-640, AES-192 için FrodoKEM-976 ve AES-256 için FrodoKEM-1344 şeklindedir (Naehrig vd, 2017; Alkim vd, 2019). Protokol genelinde kullanılmakta olan parametrelerin tanımları ve kullanılmış olan parametreler Çizelge 3.2 ve Çizelge 3.3'de detaylandırılmıştır.

Çizelge 3.2. FrodoKEM anahtar kapsülleme protokolünde kullanılan parametrelerin tanımları (Naehrig vd, 2017).

Parametre	Açıklama
$q = 2^D, D \leq 16$	Modüler indirgeme yapılan değer
$n, \bar{m}, \bar{n}$ $(n \equiv 0 \ mod \ 8)$	Tam sayı şeklindeki matris boyutları
$B \leq D$	Her matris girişinde kodlanan bit sayısı
$\ell = B \cdot \bar{m} \cdot \bar{n}$	$\bar{m} \times \bar{n}$ lik bir matriste kodlanacak bit dizilerinin uzunluğu
$len_\mu = \ell$	Mesajın bit uzunluğu
$\mathcal{M} = \{0,1\}^{len_\mu}$	Mesaj uzayı
len_A	Rastgele matris üretimi için kullanılan tohumların bit uzunluğu
len_E	Rastgele bit üretimi için kullanılan tohumların bit uzunluğu (Hata örneklemesi için)
T_χ	Örnekleme yapılırken kullanılan dağılım tablosu
len_s	Kapsülü çözme kısmında oluşabilecek bir hata için kullanılacak s bit vektörünün uzunluğu (IND - CCA güvenlik için)
len_k	Kapsülleme aşamasında oturum anahtarını üretmek için kullanılan k değerinin bit uzunluğu
len_z	A matrisini oluştururken kullanılan $seed_A$ değerini oluşturmak için kullanılan tohumların bit uzunluğu
len_{ss}	Oluşturulan oturum anahtarının bit uzunluğu
len_d	Oturum anahtarı üretilirken tuzlama (salting) için kullanılan değerin bit uzunluğu (IND – CCA güvenlik için)
$H: \{0,1\}^* \rightarrow \{0,1\}^{len_A}$	cSHAKE128$(\cdot, len_A, 0)$ veya cSHAKE256$(\cdot, len_A, 0)$ özet fonksiyonu
$G: \{0,1\}^* \rightarrow \{0,1\}^{len_E} \times \{0,1\}^{len_k} \times \{0,1\}^{len_d}$	cSHAKE128$(\cdot, len_E + len_k + len_d, 3)$ veya cSHAKE256$(\cdot, len_E + len_k + len_d, 3)$ özet fonksiyonu
$F: \{0,1\}^* \rightarrow \{0,1\}^{len_{ss}}$	cSHAKE128$(\cdot, len_{ss}, 7)$ veya cSHAKE256$(\cdot, len_{ss}, 7)$ özet fonksiyonu

Çizelge 3.3. FrodoKEM anahtar kapsülleme protokolünün uygulamasında kullanılan parametreler (Naehrig vd, 2017; Alkim vd, 2019)

Parametre	Frodo-640	Frodo-976	Frodo-1344
D	15	16	16
q	2^{15}	2^{16}	2^{16}
n	640	976	1344
$\bar{m} = \bar{n}$	8	8	8
B	2	3	4
len_A	128	128	128
$len_\mu = \ell$	128	192	256
len_E	128	192	256
len_z	128	192	128
len_s	128	192	256
len_k	128	192	256
len_d	128	192	256
len_{ss}	128	192	256
len_χ	16	16	16
χ	$\chi_{Frodo-640}$	$\chi_{Frodo-976}$	$\chi_{Frodo-1344}$
Özet Fonksiyonu	cSHAKE128	cSHAKE256	cSHAKE256

3.4.2. FrodoKEM protokolü genel yapısı

FrodoKEM protokolü; anahtar üretimi, kapsülleme ve kapsülü çözme olmak üzere üç algoritmadan oluşmaktadır. Bu algoritmaların kullanmakta olduğu bazı alt programlar bulunmaktadır. Bu kısımda, bu algoritmalarda neler yapıldığından ve bu algoritmaların kullanmakta olduğu alt programlardan bahsedilmiştir.

$Girdi: Yok$ Çıktı: $Anahtar\ Çifti\ (pk, sk')\ pk \in \{0,1\}^{len_A + D \cdot n \cdot \bar{n}}$ $sk' \in \{0,1\}^{len_S + len_A + D \cdot n \cdot \bar{n}} \times \mathbb{Z}_q^{n \times \bar{n}}$
1: $s \mid\mid seed_E \mid\mid z \leftarrow U(\{0,1\}^{len_S + len_E + len_z})$ 2: $seed_A \leftarrow H(z)$ 3: $A \in \mathbb{Z}_q^{n \times n} \leftarrow Frodo.Gen(seed_A)$ 4: S $\leftarrow Frodo.SampleMatrix(seed_E,\ n,\ \bar{n},\ T_\chi,\ 1)$ 5: E $\leftarrow Frodo.SampleMatrix(seed_E,\ n,\ \bar{n},\ T_\chi,\ 2)$ 6: $B \leftarrow AS + E$ 7: b $\leftarrow Frodo.Pack(B)$ 8: $pk \leftarrow seed_A \mid\mid b$ ve $sk' \leftarrow (s \mid\mid seed_A \mid\mid b,\ S)$

Şekil 3.1. FrodoKEM anahtar üretim algoritması (Naehrig vd, 2017)

Şekil 3.1. de görülmekte olan anahtar üretimi kısmındaki asıl önemli işlem $B \leftarrow AS + E$ işlemi ile LWE örneklerinin üretilmesi işlemdir. Bu örnekler üretilmeden önce ilk olarak rastgele ve düzgün dağılımdan sistemde kullanılacak matrislerin üretimleri için kullanılmak üzere tohumların üretimi yapılır. Devamında $H(z)$ fonksiyonu kullanılarak A matrisini oluştururken kullanılacak $seed_A$ değeri üretilir. A matrisinin üretimi için $Frodo.Gen(seed_A)$ şeklinde bir fonksiyon kullanılmaktadır. $Frodo.Gen$ fonksiyonu parametre olarak $seed_A \in \{0,1\}^{len_A}$ şeklinde bir değeri alır ve çıktı olarak rastgele $A \in \mathbb{Z}_q^{n \times n}$ matrisini verir. Bu rastgele matris üretimi AES-128 veya cSHAKE-128 algoritmaları kullanılarak yapılmaktadır (Naehrig vd, 2017). AES-128 algoritması kullanıldığında; AES algoritması 16 tane 8 bitlik blok kullandığından üretilecek matrislerin boyutlarının 16'nın katı olması gerekliliği ortaya çıkmaktadır. Matris boyutlarının 16'nın katı olmadığı durumlarda genişletme (padding) işlemi yapılarak üretilecek matrisin boyutları 16'nın katı olacak şekilde bir ayarlama yapılmalıdır. Bu ayarlamayı göz ardı edebilmek için FrodoKEM'in parametreleri 16'nın katı olarak seçilmiştir. Fakat cSHAKE algoritmasında AES algoritmasındaki gibi bir durum söz konusu değildir. Bu algoritma ile bir seferde 168 byte çıktı üretilebilmektedir. Ancak FrodoKEM'in parametrelerinin seçimi sırasında bu durum dikkate alınmamıştır. Bunun yerine fazladan üretilen bytelar atılması yolu tercih edilmiştir.

Sistemde kullanılan bir diğer matris üretim algoritması ise $Frodo.SampleMatrix()$ fonksiyonudur. Bu fonksiyon cSHAKE algoritması ve Çizelge 3.3'de verilmiş olan parametreleri kullanarak, belirli bir hata dağılımı

tablosu (T_χ) üzerinden (CDT), $n_1 \times n_2$ tane bitlerden oluşan vektör üreterek rastgele matris üretimi yapar. Anahtar üretimi kısmında kullanılan E ve S matrisleri bu fonksiyon kullanılarak üretilen matrislerdir.

Rastgele matrislerin üretimleri yapıldıktan sonra $B \leftarrow AS + E$ işlemi yapılarak LWE örnekleri ve açık anahtarın bir parçası üretilir. Buradaki $A * S$ işlemi; S matrisi sadece 8 sütun içerdiğinden matris vektör çarpım işlemi olarak düşünülebilir. $B \leftarrow AS + E$ işlemi sonucu üretilen B matrisi, $Frodo.Pack()$ fonksiyonuna parametre olarak gönderilir. Bu fonksiyon bir matrisi bit dizisine çeviren bir fonksiyondur. $Frodo.Unpack()$ fonksiyonu ise bu işlemin tam tersini yani bit dizisini matrise çevirmekte olan bir fonksiyondur (Naehrig vd, 2017). Protokolde B matrisi ve A matrisini oluştururken kullanılan $seed_A$ değeri ($seed_A \parallel B$) açık anahtar (pk), S matrisi ise gizli anahtardır (sk). Anahtar üretimi kısmında anahtar ikilisi üretildikten sonra açık anahtar kapsülleme kısmına gönderilir.

$Girdi: pk = seed_A \| B \in \{0,1\}^{len_A + D \cdot n \cdot \bar{n}}$ $Çıktı: Şifre\ metin\ c_1 \parallel c_2 \parallel d \in \{0,1\}^{(\bar{m} \cdot n + \bar{m} \cdot \bar{n})D + len_d}\ ve$ $Oturum\ anahtarı\ ss \in \{0,1\}^{len_{ss}}$
1: $\mu \leftarrow U(\{0,1\}^{len_\mu})$ 2: $seed_E \parallel k \parallel d \leftarrow G(pk \parallel \mu)$ 3: $S' \leftarrow Frodo.SampleMatrix(seed_E, \bar{m}, n, T_\chi, 4)$ 4: $E' \leftarrow Frodo.SampleMatrix(seed_E, \bar{m}, n, T_\chi, 5)$ 5: $A \leftarrow Frodo.Gen(seed_A)$ 6: $B' \leftarrow S'A + E'$ 7: $c_1 \leftarrow Frodo.Pack(B')$ 8: $E'' \leftarrow Frodo.SampleMatrix(seed_E, \bar{m}, \bar{n}, T_\chi, 6)$ 9: $B \leftarrow Frodo.Unpack(b, n, \bar{n})$ 10: $V \leftarrow S'B + E''$ 11: $C \leftarrow V + Frodo.Encode(\mu)$ 12: $c_2 \leftarrow Frodo.Pack(C)$ 13: $ss \leftarrow F(c_1 \parallel c_2 \parallel k \parallel d)$ 14: $Çıktı\ şifre\ metin\ c_1 \parallel c_2 \parallel d\ ve\ \ oturum\ anahtarı\ ss$

Şekil 3.2. FrodoKEM kapsülleme algoritması (Naehrig vd, 2017)

Kapsülleme aşamasında rastgele bir bit dizisi olan μ ve şifre metinleri üretirken kullanılacak olan rastgele değerler üretilir (k, d). Devamında S', E' ve E'' matrisleri belirli bir dağılım tablosu kullanılarak üretilir. A matrisi bu kısımda $Frodo.Gen()$ fonksiyonu ile tekrar üretilir ve $B' \leftarrow S'A + E'$ işlemi ile birlikte, şifre metnin ilk parçasını oluşturmak için kullanılır. Açık anahtarı oluşturan ikiliden B matrisi ise $V \leftarrow S'B + E''$ işlemi ile şifre metnin ikinci parçasını oluşturmak için kullanılır.

Buradaki $S' * A$ ve $S' * B$ işlemleri matris vektör çarpım işlemi olarak düşünülebilir. Ek olarak kapsülü çözme aşamasında kullanılacak olan μ değeri V matrisine eklenir ve C matrisi oluşturulur. Şifre metinler için elde edilen B' ve C matrisleri $Frodo.Pack()$ fonksiyonu ile bit dizisine dönüştürülerek c_1 ve c_2 şifre metin parçaları elde edilir. Devamında şifre metin parçalarına tuzlama (salting) işlemi yapılır ve elde edilen değerin özetinin alınması ile oturum anahtarı elde edilmiş olur. Tuzlama işlemi özet fonksiyonlarda güvenliği artırmak için kullanılmakta olan bir yöntemdir (Boneh ve Shoup, 2015). Rastgele bir değer üretilir (protokolde bu değer d değişkenidir) ve özeti alınacak değere eklenir. Özet fonksiyonlarda, özeti alınacak değer üzerinde 1 bitlik bir değişim yapılsa bile bu değişim çıkacak sonucu büyük oranda etkilemektedir. Bu durumdan dolayı her oturum anahtarı oluşturulması işleminde; şifre metinler aynı olsa bile, eklenen salt değeri farklı olacağından oluşacak çıktı farklı olacaktır ve bu durum sistemin güvenliğini artırmaktadır. Kapsülleme aşamasında son olarak oluşturulan şifre metinler kapsülü çözme aşamasına gönderilir.

$Girdi: Şifre\ metin\ c_1 \| \| c_2 \| \| d \in \{0,1\}^{(\bar{m}\cdot n+\bar{m}\cdot\bar{n})D+len_d}$, $sk' = (s \parallel seed_A \parallel b, S) \in \{0,1\}^{len_S+len_A+D\cdot n\cdot\bar{n}} \times \mathbb{Z}_q^{n\times\bar{n}}$ $Çıktı: Oturum\ anahtarı\ ss \in \{0,1\}^{len_{ss}}$
1: $B' \leftarrow Frodo.Unpack(c_1)$ 2: $C \leftarrow Frodo.Unpack(c_2)$ 3: $M \leftarrow C - B'S$ 4: $\mu' \leftarrow Frodo.Decode(M)$ 5: $pk \leftarrow seed_A \parallel b$ 6: $seed'_E \parallel k' \parallel d' \leftarrow G(pk \parallel \mu')$ 7: $S' \leftarrow Frodo.SampleMatrix(seed_E, \bar{m}, n, T_\chi, 4)$ 8: $E' \leftarrow Frodo.SampleMatrix(seed_E, \bar{m}, n, T_\chi, 5)$ 9: $A \leftarrow Frodo.Gen(seed_A)$ 10: $B'' \leftarrow S'A + E'$ 11: $E'' \leftarrow Frodo.SampleMatrix(seed_E, \bar{m}, \bar{n}, T_\chi, 6)$ 12: $B \leftarrow Frodo.Unpack(b, n, \bar{n})$ 13: $V \leftarrow S'B + E''$ 14: $C' \leftarrow V + Frodo.Encode(\mu')$ 15: Eğer $B' \parallel C = B'' \parallel C''$ ve $d = d'$ ise 16: $\quad$ $çıktı\ ss \leftarrow F(c_1 \parallel c_2 \parallel k' \parallel d)$ 17: Aksi halde   18: $\quad$ $çıktı\ ss \leftarrow F(c_1 \parallel c_2 \parallel s \parallel d)$

Şekil 3.3. FrodoKEM kapsülü çözme algoritması (Naehrig vd, 2017)

Kapsülü çözme aşamasında ilk olarak c_1 ve c_2 şifre metinleri (bit dizileri) $Frodo.Unpack()$ fonksiyonu kullanılarak B' ve C matrislerine dönüştürülür. Devamında elde geçerli bir şifre metin olup olmadığını kontrol etmek amacıyla μ değerinin şifresi çözülür. A, S', E' ve E'' matrisleri tekrar üretilerek kapsülleme aşamasında üretilen şifre metinler, kapsülleme aşamasındaki işlemler tekrarlanarak üretilmeye çalışılır. Eğer üretilen şifre metinler öncekiler ile aynı şekilde üretilmiş ise oturum anahtarı, şifre metinler ve salt değerinin özeti alınarak tekrar oluşturulur ve paylaşılır (Naehrig vd, 2017; Howe vd, 2018). Bu kısımda B' ve C matrisleri kapsülleme aşamasındaki şekilde üretildiyse, k' değeri direk oturum anahtarını üretmek için kullanılır. Üretilen B' ve C matrisleri kapsülleme aşamasındakinden farklı ise oturum anahtarını üretmek için k' yerine rastgele bir değer gönderilir. Bu şekilde, bir saldırı durumunda saldırgan hata yapılıp yapılmadığını öğrenemez. Bu işlemin amacı IND-CCA güvenliği sağlamaktır.

Çizelge 3.4. Bir anahtar kapsülleme protokolü için IND-CCA oyun tanımı (Jiang vd, 2018)

IND-CCA Oyunu	Kapsülü Çözme(sk, c)
1: $(pk, sk) \leftarrow Anahtar\ Üretimi$	1: $Eğer\ c = c^*$
2: $b \xleftarrow{\$} \{0,1\}$	2: $\quad Geri\ döndür\ \perp$
3: $(K_0^*, c^*) \leftarrow Kapsülü\ Çözme(pk)$	3: $Aksi\ halde\ geri\ döndür$
4: $K_1^* \xleftarrow{\$} K$	4: $\quad K := Kapsülü\ Çözme(sk, c)$
5: $b' \leftarrow A^{Kapsülleme}(pk, c^*, K_b^*)$	
6: $Geri\ döndür\ b' =? b$	

IND-CCA güvenliği; Fujisaki-Okamoto dönüşümü ile sağlanmaktadır (Fujisaki ve Okamoto, 1999). Bu dönüşüm; klasik rastgele kehanet modeli ile IND-CPA bir açık anahtarlı şifreleme düzeninden, IND-CCA güvenliğe sahip açık anahtarlı şifreleme şeması oluşturan bir sistemdir. FrodoKEM protokolünde; IND-CPA güvenliğe sahip açık anahtarlı bir sistemden, IND-CCA güvenliğe sahip açık anahtarlı bir sisteme dönüşüm işlemi bulunmaktadır. (Naehrig vd, 2017). Bu dönüşüm işlemleri genellikle protokollerde IND-CCA oyunu olarak ifade edilmektedir ve Çizelge 3.4'de gösterilmektedir.

KEM güvenliğini belirlemek için açık anahtar şifreleme sistemleri üzerinden, oyun tabanlı güvelik kavramları KEM'lerle çalışmak üzere uyarlanmıştır. Oyuna dayalı bir tanım genellikle, söz konusu uygulamaya karşı belirli bir saldırı başlatan bir saldırgan varlığında korunması gereken bir güvenlik özelliği ile tanımlanmaktadır.

Şeçilmiş şifreli metin saldırısı (CCA), kriptanalistin seçilen şifreli metinlerin şifresini çözerek bilgi toplayabileceği bir saldırı modelidir. Bu bilgi parçalarından, saldırgan taraf şifre çözmek için kullanılan gizli anahtarı elde etmeyi deneyebilir. Bu saldırı temel olarak şu şekilde gerçekleştirilir (Bogdanov, 2005):

1. Meydan okuyan açık anahtarı (PK) ve gizli anahtarı (SK) belirli bir güvenlik parametresine (k) göre üretir, PK yı saldırgan tarafa verir ve SK yı saklar
2. Saldırgan istediği sayıda şifreleme işlemi yapabilir ve isteğe bağlı şifreleme metinlerine dayalı şifre çözme kodlarına çağrı yapabilir.
3. Saldırgan taraf iki ayrı seçilmiş açık metni gönderir (M0, M1).
4. Meydan okuyan taraf rastgele bir bit (b) seçer ve meydan okuma olarak şifre metni saldırgana geri gönderir E(PK, Mb).
5. Uyarlanamayan (non-adaptive) durumda (IND-CCA) saldırgan taraf şifre çözme kahini için başka çağrı yapamaz. Uyarlanabilir (adaptive) durumda (IND-CCA2) şifre çözme kahini için çağrı yapabilir fakat şifre metni meydan okuması yapmayabilir.
6. Son olarak saldırgan taraf tahmin ettiği b değerini çıktı verir.

Bu saldırıdan kaçınmak için şifreli metin dizgisinin, rastgele bir dizgiden ayırt edilemez olduğu şifreleme şemalarına ihtiyaç olmaktadır. Şifreli iletişim kurulurken, trafik analizini zorlaştırmak için her şifrelenmiş verinin içeriğinin rastgele bir veriden ayırt edilemez hale getirilmesi tercih edilmektedir.

4. MATRİS VEKTÖR ÇARPIM KÜTÜPHANESİ

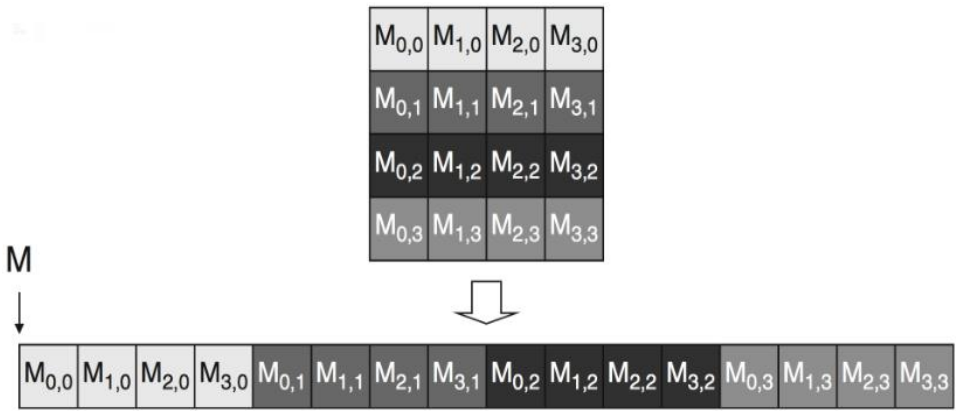
Önceki bölümlerde anlatılan FrodoKEM protokolünde ve LWE problemini kullanan diğer anahtar kapsülleme protokollerinde matris vektör çarpım işlemleri rastgele sayı üretiminden sonra en çok zaman harcayan işlemdir. Protokollerin içerisinde çok sayıda matris vektör çarpım işlemi bulunduğundan ve kullanılan matrislerin boyutları büyük olduğundan bu çarpım işlemlerinin hızlı ve verimli yapılması protokollerin verimlilikleri açısından büyük önem taşımaktadır. Bu durumdan yola çıkarak, tez kapsamında FrodoKEM protokolünde kullanılan matris vektör çarpım yöntemleri, genelleştirilmeye çalışılarak bir matris vektör çarpım kütüphanesi oluşturulması amaçlanmıştır. Bu kütüphane; kafes tabanlı bir kriptosistem oluşturmak isteyen kişilerin tanımlanmış olan matris vektör çarpım fonksiyonlarını sadece matris boyutu değiştirerek kullanabilmeleri amacıyla oluşturulmuştur.

Kütüphanenin oluşturulma aşamasında ilk olarak FrodoKEM-640 protokolü içerisindeki matris vektör çarpımları incelenmiştir. FrodoKEM protokolü içerisinde üç çeşit çarpım işlemi bulundurmaktadır. Bunlar; $A * S, S' * A$ ve $S' * A$ çarpımının vektör komutları ile yapılmış halidir. Burada A matrisi $m \times n$ (m > n) boyutunda dikdörtgen şeklinde bir matris veya $n \times n$ boyutunda kare şeklinde bir matris olabilmektedir. S matrisi ise çarpım işlemi içerisinde vektörü temsil etmektedir. FrodoKEM-640 protokolü içerisindeki matris vektör çarpım işlemleri A matrisinin 640×640 ve S matrisinin 640×8 boyutlarında olduğu durumlarda düzgün çalışmaktadır (Naehrig vd, 2017). Oluşturulmuş olan çarpım kütüphanesi bu çarpım yöntemlerinin kullanılan matrislerin boyutu ve şekli fark etmeksizin çalışabilecek şekilde otomatikleştirilmiş bir modelidir.

Çarpım işlemlerinin verimliliğini artırmak için kullanılacak veriler çarpım işlemleri yapılmadan önce hafızaya hizalanarak yerleştirilmektedir. Bu hizalama işlemini amacı, ön belleği en verimli şekilde kullanmak ve çarpım işlemini hızlandırıp verimini artırmaktır. Ön bellek; işlemcide gerçekleştirilen hesaplamalar için sıklıkla ihtiyaç duyulan geçici verileri tutan, işlemci ile ana bellek arasında bulunan çok yüksek hızda özel bir bellek çeşididir (Mano, 2005). İşlemcinin ana belleğe erişim hızı önbelleğe erişim hızından düşük olduğundan önbellek programların gerçekleştirilmesi sırasındaki hız kaybını telafi etmek ve programların çalışma sürelerini hızlandırmak için kullanılmaktadır (Mano, 2005).

Önbelleği en verimli şekilde kullanmak için verilerin hizalanması, program parçalarını verimli hale getirmek için çok kullanılan bir yöntemdir. Bu işlem yapılırken kullanılacak verinin adresi, ön bellek blok boyutunun bir katı olmalıdır. Ön bellek blok boyutu; ön belleğin her bir bloğuna bir seferde kaç tane bitişik byte alınacağını ifade eden bir değerdir (Lebeck ve Wood, 1994). Örnek olarak tezde kullanılan bilgisayarın ön bellek blok boyutu 64 byte'dır. Statik olarak tanımlanan veriler genellikle derleyici desteği ile bu hizalamayı yapmaktadır. Fakat dinamik tanımlanan veriler, işaretçiler kullanılarak istenilen şekilde düzenlenip hizalanabilir. Örnek olarak C programlama dili için; malloc() fonksiyonu kullanılarak hafızada yer ayrılan veriler, ön bellek bloğunda hizalanmış verilerin kullanılmasına olanak tanır (Lebeck ve Wood, 1994).

Program parçaları hazırlanırken kullanılacak verilerin en verimli şekilde kullanılabilmesi için, verilerin hafızaya yerleşimlerinin kullanılacak veriye uygun bir şekilde yapılması gerekmektedir. Ana bellek (RAM) doğrusal bir adres alanına sahiptir ve bu tek boyutlu bir dizi olarak düşünülebilir. Bu durum, yapılan uygulama iki boyutlu bir diziye sahip olduğunda (matris), derleyicinin bir şekilde bu dizinin iki boyutunu düzgün depolayabilmek için hafızanın tek boyutuyla eşleştirmesi gerekliliğini ortaya çıkarmaktadır (Row Major Ordering). Bunu yapmanın iki yolu vardır. Bunlar; satır tabanlı düzen (row - major order) ve sütun tabanlı düzendir (column - major order) (Lexicographic Array Storage). $M \times N$' lik iki boyutlu bir A dizisi için, bu yerleşim şekilleri ile A dizisinin i. satır ve j. sütunundaki elemanın adresi $A^{(M,N)}_{satır-tabanlı}(i,j) = N \times i + j$ ve $A^{(M,N)}_{sütun-tabanlı}(i,j) = M \times j + i$ şeklinde gösterebilir (Thiyagalingam vd, 2003).



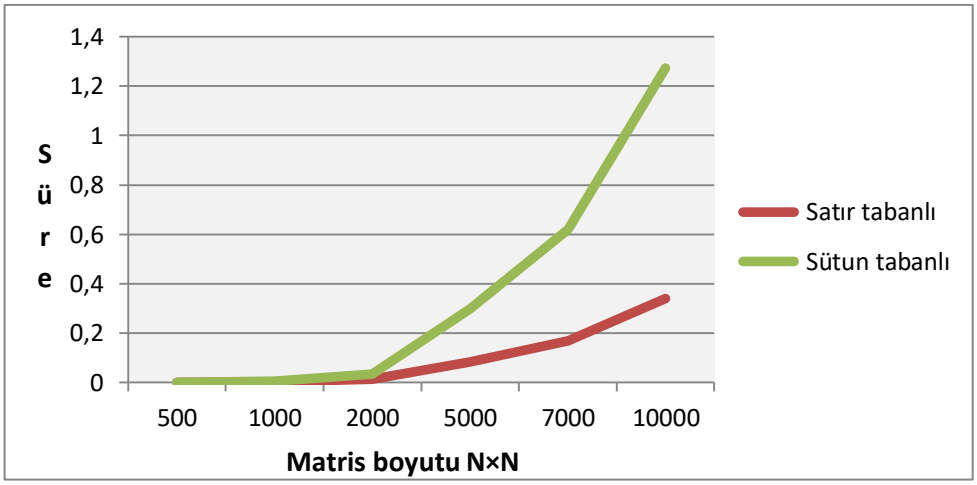
Şekil 4.1. Satır tabanlı düzen (Flatten 2D matrix)

İşlemciler hafızadaki veriler arasında zıplamalar yerine, doğrusal bir şekilde sırayla verilere eriştiğinde daha verimli çalışmaktadırlar. Bu durumdan dolayı; verileri hafızadan alıp kullanırken satır tabanlı düzen kullanıldığında işlemciden daha fazla verim alınmaktadır. C programlama dili verileri kullanırken satır tabanlı düzen kullanmaktadır. Dolaylı olarak FrodoKEM protokolü ve matris vektör çarpım kütüphanesinde satır tabanlı düzen yöntemi kullanılmıştır. Şekil 4.2'de bu yerleşim düzenleri için basit bir C kodu verilmiştir.

```c
int matris[m][n];

//Satır tabanlı düzen

for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
        matris[i][j] += j;

//Sütun tabanlı düzen

for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
        matris[j][i] += j;
```

Şekil 4.2. Satır tabanlı düzen ve sütun tabanlı düzen için örnek C kodu

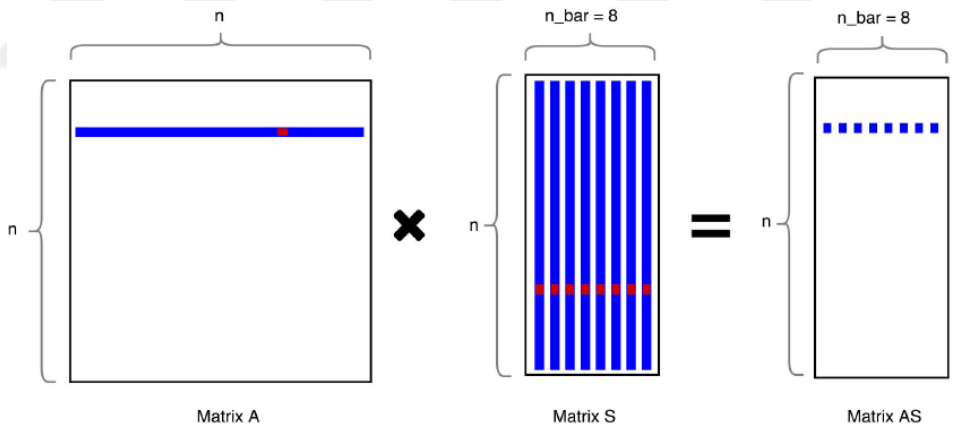
Şekil 4.3'de görüldüğü üzere; veri olarak iki boyutlu diziler kullanıldığında, satır tabanlı düzen yöntemi sütun tabanlı düzen yönteminden daha verimli ve hızlı çalışmaktadır (Sonuçlar için Şekil 4.2 deki kod parçası kullanılmıştır).



Şekil 4.3. N×N boyutunda matrisler için satır tabanlı düzen ve sütun tabanlı düzen işlem süreleri (süreler saniye cinsindendir)

4.1. $A * S$ Çarpımı

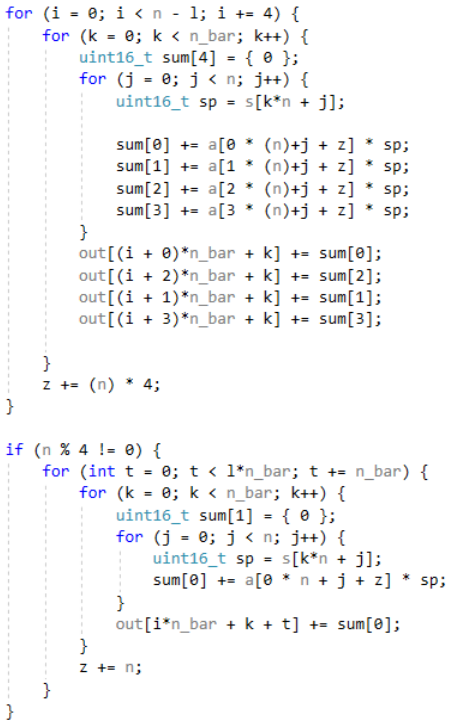
LWE tabanlı protokollerde LWE örneklerini üretmek için $A * S$ çarpımı yapılmaktadır. Oluşturulan kütüphane içerisindeki bu çarpım şeklinde A matrisinden her seferinde dörder satır alınarak S matrisi ile çarpımı yapılmaktadır. FrodoKEM yazarlarına göre; A matrisinden her seferinde dört satırı S matrisi ile çarpmak bu sistem için en verimli sonucu üretmektedir (Naehrig vd, 2017). Çarpım işlemi için A matrisi üretilirken önceki bölümde anlatıldığı gibi hafızaya hizalanarak yerleştirilmektedir. Rastgele A matrisinin üretimi AES veya cSHAKE algoritmaları kullanılarak yapılmaktadır (Genellikle AES kullanılmıştır). S matrisi rastgele bir matris olduğundan transpozu alınmış şekilde üretildiği varsayılmıştır. Bu varsayımın sebebi ise verilere satır tabanlı düzen ile erişim şeklinin maliyetinin daha düşük olmasından dolayıdır. S matrisi transpoz olarak varsayıldığında; S matrisinden çarpma işlemine katılacak elemanlar hafızada doğrusal bir şekilde yerleşmektedir. Satır tabanlı düzen ve transpoz halindeki S matrisi ile birlikte çarpma işlemi, bu durumun tersine oranla çok daha verimli ve hızlı olmaktadır.



Şekil 4.4. A * S çarpımına FrodoKEM üzerinden örnek bir şekil (Howe vd, 2018)

Oluşturulan A matrisinin satır sayısı 4'e tam bölünebilen bir sayı ise çarpım işlemi, bir döngü aracılığıyla her adımda A matrisinden dört satırı S matrisi ile çarparak ilerlemektedir. Eğer A matrisinin satır sayısı 4'e tam bölünmeyen bir sayı ise; artan satırlar tek tek S matrisi ile çarpılmaktadır. Tek tek bu çarpımlar yapıldıktan sonra sonuca eklenmesi işlemi yapılmaktadır. LWE probleminin tanımında olduğu gibi bu çarpım işlemine, ayrık Gauss dağılımı ile önceden üretimi yapılmış olan hata matrisi (E) eklenmesi işlemi çarpım işleminden önce yapılmaktadır. Yani $A * S$ çarpımından elde edilen sonuç hata matrisinin üzerine

toplanarak LWE örnekleri oluşturulmaktadır. Bu çarpım yöntemi derleyici (gcc) tarafından otomatik olarak paralelleştirilebilen bir yapıya sahiptir. Bu durumdan dolayı hızlı ve verimli çalışmakta olan bir yöntemdir.

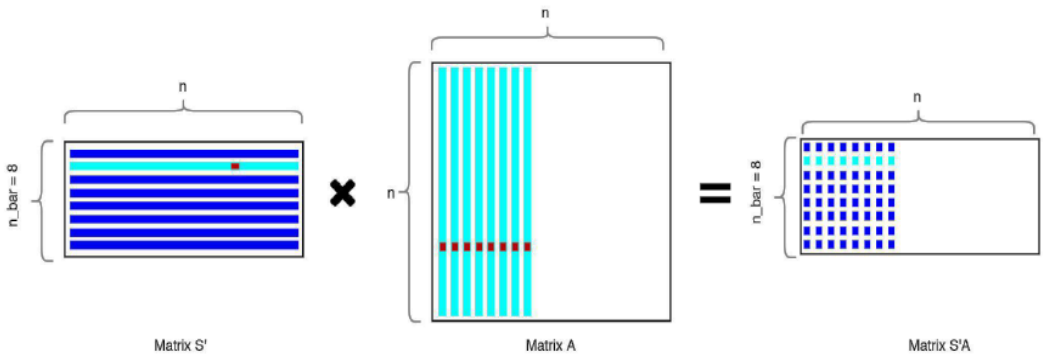


```
for (i = 0; i < n - l; i += 4) {
    for (k = 0; k < n_bar; k++) {
        uint16_t sum[4] = { 0 };
        for (j = 0; j < n; j++) {
            uint16_t sp = s[k*n + j];

            sum[0] += a[0 * (n)+j + z] * sp;
            sum[1] += a[1 * (n)+j + z] * sp;
            sum[2] += a[2 * (n)+j + z] * sp;
            sum[3] += a[3 * (n)+j + z] * sp;
        }
        out[(i + 0)*n_bar + k] += sum[0];
        out[(i + 2)*n_bar + k] += sum[2];
        out[(i + 1)*n_bar + k] += sum[1];
        out[(i + 3)*n_bar + k] += sum[3];

    }
    z += (n) * 4;
}

if (n % 4 != 0) {
    for (int t = 0; t < l*n_bar; t += n_bar) {
        for (k = 0; k < n_bar; k++) {
            uint16_t sum[1] = { 0 };
            for (j = 0; j < n; j++) {
                uint16_t sp = s[k*n + j];
                sum[0] += a[0 * n + j + z] * sp;
            }
            out[i*n_bar + k + t] += sum[0];
        }
        z += n;
    }
}
```

Şekil 4.5. A ∗ S çarpımı C kodu

4.2. $S' * A$ Çarpımı

Bu çarpım şekli; FrodoKEM protokolünde anahtar kapsülleme kısmında şifre metinleri elde etmek için kullanılmakta olan bir yöntemdir. Önceki bölümlerde anlatıldığı gibi bu çarpma şeklinde de S′ ve A matrisleri satır tabanlı düzen ile birlikte kullanılmaktadır. Bu çarpım yönteminde oluşturulan A matrisi transpoz olarak kullanılmaktadır. Fakat bu varsayım olarak değildir. A matrisinin transpozu çarpma işleminden önce alınmaktadır ve A matrisi AES algoritması kullanılarak üretilmektedir. Bu durum yine satır tabanlı düzen ile birleştirildiğinde çarpma işleminin verimliliği artmaktadır.



Şekil 4.6. $S' * A$ çarpımına FrodoKEM üzerinden örnek bir şekil (Howe vd, 2018)

$S' * A$ çarpımında A matrisinden bir döngü aracılığıyla her seferinde sekiz tane sütun alınarak S' matrisi ile çarpımı yapılmaktadır. A matrisini sütun sayısı sekize tam bölünebilen bir sayı ise çarpma işlemi her seferinde sekiz tane sütunu çarparak devam etmektedir. Eğer A matrisinin sütun sayısı sekize tam bölünmeyen bir sayı ise, artan sütunlar S' matrisi ile tek tek çarpılarak sonuca eklenmektedir.

```
    for (i = 0; i < nbarbar; i++) {
        for (k = 0; k < PARAMS_STRIPE_STEP; k += PARAMS_PARALLEL) {
            uint16_t sum[PARAMS_PARALLEL] = { 0 };
            for (j = 0; j < n; j++) {
                uint16_t sp = s[i*n + j];

                sum[0] += sp * a_t[(k + 0)*(n)+j];
                sum[1] += sp * a_t[(k + 1)*(n)+j];
                sum[2] += sp * a_t[(k + 2)*(n)+j];
                sum[3] += sp * a_t[(k + 3)*(n)+j];
            }
            out[i*(nbar)+kk + k + 0] += sum[0];
            out[i*(nbar)+kk + k + 1] += sum[1];
            out[i*(nbar)+kk + k + 2] += sum[2];
            out[i*(nbar)+kk + k + 3] += sum[3];
        }
    }
    clear_words(A, n * PARAMS_STRIPE_STEP / 2);
    clear_words(a_t, n * PARAMS_STRIPE_STEP / 2);
}
k = n - (n % 8);
if (n % 8 != 0) {
    for (int t = 0; t < l; t += 1) {
        k = k + t;
        for (i = 0; i < nbarbar; i++) {
            uint16_t sum[1] = { 0 };
            for (j = 0; j < n; j++) {
                uint16_t sp = s[i*n + j];
                sum[0] += sp * a_t[j + z];
            }
            out[k] += sum[0];
            k += n;
        }
        k = n - (n % 8);
        z += n;
    }
}
```

Şekil 4.7. $S' * A$ çarpımı C kodu

Çarpımdan gelen sonuca hata matrisi eklenmesi işlemi, önceki çarpım yönteminde olduğu gibi yine çarpım işleminden önce yapılmaktadır. Bu çarpım şekli derleyici tarafından otomatik olarak paralelleştirilemeyen bir yapıya sahiptir. Bu

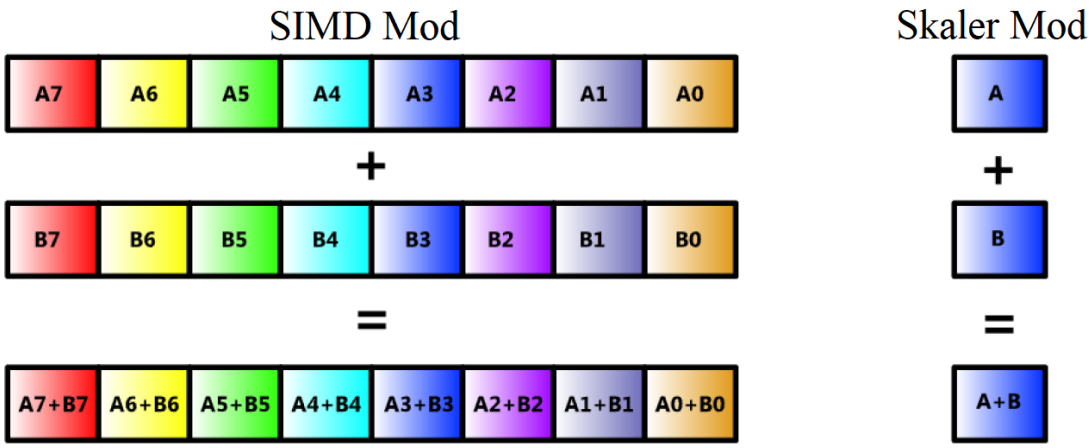
durumdan dolayı uygulamada $A * S$ çarpımına oranla daha düşük verimle çalışmaktadır. Bu verimi artırmak adına bu çarpım yönteminin vektör komutları ile yapılmış hali FrodoKEM ve oluşturulmuş olan matris vektör çarpım kütüphanesinde bulunmaktadır.

4.3. $S' * A$ Vektör Komutları ile Çarpımı

Vektörleştirme işlemi uygulamalar üzerinde basit olarak; bir döngüyü yeniden yazma işlemidir. Vektörleştirme; çok büyük verilerin verimli bir şekilde işlenmesi gereken hesaplamalarda sıkılıkla kullanılmaktadır. Bu yöntem; N boyutlu bir dizinin elemanları üzerinde herhanagi bir işlem (çarpma, toplama gibi) yapılmak istendiğinde bu işlemi dizinin bütün elemanları üzerinde tek tek yapmak yerine, dizinin elemanlarını bir vektörün elemanları olarak yerleştirip eş zamanlı olarak yapılmasını sağlamakta olan bir yöntemdir (The significance of SIMD). Bu işlem yapılırken FrodoKEM protokolünde ve bu tez kapsamında Intel AVX2 (Advance Vector Extensions) komut seti içerisinde tanımlanmış olan kodlar kullanılmıştır. Bu komut seti Intel işlemciler üzerinde SIMD (Single Instruction Multiple Data - Çok Veri Tek Komut) denilen işlemleri daha hızlı, verimli yapmak adına oluşturulmuş olan bir sistemdir. SIMD komutları tek adımda birden fazla verinin işlenmesini sağlamakta olan komutlardır (Lomont, 2011). Bunu sağlarken önceki komut setleri üzerinde bazı değişiklikler yapılmıştır. Bu değişikliklerden bazıları (Lomont, 2011);

- 128 bitlik yazmaçların (register) 256 bitlik yazmaçlara genişletilmesi,
- Bazı gereksiz komutların kaldırılması ile daha basit ve hızlı bir komut seti elde edilmesi,
- Kullanılacak veriler hafızada hizalanırken daha esnek bir yapı olması

gibi değişikliklerdir. Eski komut setlerinde kullanılan skaler yöntem, verileri tek tek ele alarak çalışmaktadır. Vektör komutları ise hafızadaki verileri paralel ele alarak çalışır. Bu sayede vektör komutları hafıza içerisinde daha az veri hareketi sağlayarak skaler yönteme oranla daha verimli çalışmaktadır. Ek olarak matris çarpım kütüphanesi içerisindeki diğer çarpım yöntemleriyle kıyaslandığında; vektör komutları kullanılan çarpım yöntemi daha verimli ve hızlı çalışmaktadır.



Şekil 4.8. 32 bitlik veriler ile işlem yapılırken, verilerin vektör komutları ve tek tek alınması arasındaki fark (Lomont, 2011)

Matris vektör çarpım kütüphanesinde kullanılan bu yöntem içerisinde 256 bitlik vektörler ve 16 bitlik tamsayılar kullanılmıştır. Bu sayede bir seferde 16 tane tam sayı bir vektörün içine yerleştirilebilmekte ve aynı anda 16 sayının çarpımı yapılmaktadır. A ve S' matrisleri üretildikten sonra A matrisinin transpozu alınmaktadır. Bu işlemin amacı; Bölüm 4'ün başında anlatılıdığı üzere hafızada yapılacak atlamaları engelleyip hafızaya doğrusal bir şekilde erişim sağlamaktır. Bu yöntemde A matrisinden bir seferde 16 tane (256 bit) eleman alındığı için, A matrisinin sütun sayısının 16'ya tam bölünebilen bir sayı olup olmadığının kontrolü yapılmaktadır.

```
for (k = 0; k < PARAMS_NBAR; k++) {
  ALIGN_HEADER(32) uint32_t sum0[8], sum1[8], sum2[8], sum3[8] ALIGN_FOOTER(32);
  __m256i a0, a1, a2, a3, b, acc0, acc1, acc2, acc3;
  acc0 = _mm256_setzero_si256();
  acc1 = _mm256_setzero_si256();
  acc2 = _mm256_setzero_si256();
  acc3 = _mm256_setzero_si256();
  b = _mm256_setzero_si256();
  for (j = 0; j < PARAMS_N; j += 16) {
    b = _mm256_load_si256((__m256i*)&s[k*PARAMS_N + j]);
    a0 = _mm256_load_si256((__m256i*)&a_row[(0*PARAMS_N) + j]);
    a0 = _mm256_madd_epi16(a0, b);
    acc0 = _mm256_add_epi32(a0, acc0);
    a1 = _mm256_load_si256((__m256i*)&a_row[(1*PARAMS_N) + j]);
    a1 = _mm256_madd_epi16(a1, b);
    acc1 = _mm256_add_epi32(a1, acc1);
    a2 = _mm256_load_si256((__m256i*)&a_row[(2*PARAMS_N) + j]);
    a2 = _mm256_madd_epi16(a2, b);
    acc2 = _mm256_add_epi32(a2, acc2);
    a3 = _mm256_load_si256((__m256i*)&a_row[(3*PARAMS_N) + j]);
    a3 = _mm256_madd_epi16(a3, b);
    acc3 = _mm256_add_epi32(a3, acc3);
  }
}
```

Şekil 4.9. $S' * A$ vektör komutları ile çarpımı kodları (kodların sadece bir kısmı paylaşılmıştır)

Eğer sütun sayısı 16'nın katı ise; çarpım işlemi bir döngü aracılığı ile A matrisinin 16 sütununu vektör komutlarını kullanarak S' matrisi ile çarpmaktadır. Bu işlem _mm256_madd_epi32 komutu ile yapılmaktadır. A matrisinin sütun sayısı 16'ya bölünmeyen bir sayı ise artan sütunlar yine vektör komutları kullanılarak tek tek S' matrisi ile çarpılıp sonuca eklenmektedir. Bu yöntem şu an için sadece A matrisinin kare matris olduğu durumlarda çalışmakta olan bir yöntemdir.

5. PROTOKOLLERİN UYGULAMALARI ÜZERİNDE YAPILAN DEĞİŞİKLİKLER

Bu tez kapsamında incelenmiş olan anahtar kapsülleme protokolleri teoride aynı işlemleri yapıyor olsalarda, protokollerin uygulamalarında farklılıklar bulunmaktadır. Bu farklılıklar protokollerin uygulamaların çalışma süreleri üzerinde kötü etki yaratan durumlara sebep olabilmektedir. Bu farkılıklıklardan tez kapsamında incelenenler şunlardır;

- Matris vektör çarpım işlemleri; FrodoKEM protokolünde önceki bölümlerde anlatılıdığı gibi verimli bir çarpım algoritması kullanılırken, Lizard, Emblem ve Lotus protokollerinde standart matris çarpım algoritması kullanılmaktadır. Standart matris çarpım algoritması protokollerin yavaş çalışmasına sebep olmaktadır. Bu kısım rastgele sayı üretiminden sonra protokollerde en çok zaman almakta olan kısımdır.
- Protokollerde kullanılan matrislerin şekilleri ve boyutları; FrodoKEM ve Lotus protokollerinde A matrisi kare matris şeklinde iken, Lizard ve Emblem protokollerinde A matrisi dikdörtgen şeklindedir.
- Rastgele sayı üretim fonksiyonları; FrodoKEM protokolünde rastgele sayı üretimi için AES veya cSHAKE algoritmaları kullanılırken, Lizard ve Lotus protokollerinde sözde rastgele sayı üreticileri kullanılmaktadır. Emblem protokolünde ise rastgele sayı üretimi için C programlama dili içerisindeki "Random" kütüphanesinin fonksiyonları kullanılmaktadır. Rastgele sayı üretimi protokollerde en çok zaman almakta olan kısımdır.
- LWE örneklerinin üretimi; FrodoKEM ve Emblem protokollerinde LWE örnekleri $A * S + E$ şeklinde üretilirken, Lizard ve Lotus protokollerinde ise $-A * S + E$ şeklinde LWE örneği üretimi yapılmaktadır.
- Protokollerde indirgeme yapılan değerler; bu değerler parametre setlerine göre farklılıklar göstermektedir.

Bu farklılıklardan bazılarının protokollerin çalışma zamanı üzerinde bir etkisi olmazken, matris çarpımı ve rastgele sayı üretimi gibi farklılıklar protokollerde çalışma zamanı açısından fark yaratmaktadır. Matris vektör çarpım kütüphanesi içerisindeki çarpımların bu protokollere yerleştirilme işleminden önce protokollerin genel yapıları, kullanmakta oldukları parametreler ve protokollerde kullanılan matris

vektör çarpım işlemleri incelenmiştir. Yapılmış olan matris çarpım kütüphanesi protokollere entegre edilirken yukarıda anlatılan farklılıklar göz önüne alınmıştır ve bu farklılıklara göre uygulamalarda değişiklikler yapılmıştır. Burada dikkat edilmesi gereken bir nokta A matrisidir. Oluşturulmuş olan protokollerde sistemin güvenliğini doğrudan etkileyen faktörlerden birisi A matrisinin sütun sayısıdır. Bu yüzden parametre setleri A matrisinin sütun sayısı ile ifade edilmektedir. Çizelge 5.1'de tez kapsamında incelenen protokollerin parametre setlerinin NIST projesi kapsamındaki güvenlik seviyelerinin karşılıkları verilmiştir.

Çizelge 5.1. Tez kapsamında incelenen protokollerin parametre kümeleri ve karşılık gelen güvenlik seviyeleri

Protokol	Boyut	Mod Değeri	Güvenlik	Kategori
FrodoKEM	N = 640	log q = 15	AES128	1
	N = 976	log q = 16	AES192	3
	N = 1344	log q = 16	AES256	5
Lizard	N = 536	log q = 11	AES128	1
	N = 663	log q = 10	AES128	1
	N = 816	log q = 11	AES192	3
	N = 952	log q = 11	AES192	3
	N = 1088	log q = 12	AES256	5
	N = 1300	log q = 11	AES256	5
Emblem	N = 611	log q = 24	AES128	1
	N = 770	log q = 24	AES128	1
Lotus	N = 576	log q = 13	AES128, SHA256	1, 2
	N = 704	log q = 13	AES192, SHA384	3, 4
	N = 832	log q = 13	AES256	5

Matris vektör çarpım işlemlerinin protokollerde sorunsuz çalışması için protokollerdeki parametreler ve protokollerin genel işleyişlerine göre çarpım işlemleri düzenlenmiştir. Protokollerde değiştirilen çarpım işlemleri yerine, matris vektör çarpım kütüphanesinden $A * S$ veya $S' * A$ çarpımları kullanılmıştır. Vektör işlemleri ile yapılan çarpım işlemleri kullanılmamıştır. Fakat vektör işlemleri ile

yapılan çarpım kullanıldığı takdirde bu işlemler bir seferde 256 bitlik vektörleri çarptığından diğer yöntemlere oranla çok daha hızlı çalışacağı öngörülmektedir.

Bu bölümün devamında Lizard, Emblem ve Lotus protokollerinin genel yapıları ve protokoller üzerinde yapılan değişikliklerle birlikte elde edilen sonuçlardan bahsedilmiştir. Lizard protokolü için ek olarak, protokolün açık kaynaklı farklı bir uygulaması üzerinde yapılan bir yan kanal saldırısı ile ilgili çalışma temel alınarak, uygulamanın güvenliğini artırmak için yazılım üzerinde yapılan değişikliklerden bahsedilmiştir. Son olarak ise bu protokollerin FrodoKEM benzeri uygulamalarının yapımı için FrodoKEM protokolünde önerilen yöntemlere ek olarak uygulamada yapılan değişikliklerden bahsedilmiştir. Yapılan bütün değişiklikler, Emblem protokolü hariç, protokollerin referans uygulamaları (Referance Implementation) üzerinde yapılmıştır. Emblem protokolünde ise yapılan değişiklikler optimize edilmiş (Optimized Implementation) uygulama üzerinde yapılmıştır. Yapılan değişiklikler protokollerin kendileri üzerinde değil uygulamaların üzerinde yapılan değişikliklerdir.

5.1. Lizard Protokolünde Yapılan İyileştirmeler ve Elde Edilen Sonuçlar

Lizard protokolü, NIST'e önerilmiş olan ve standart LWE problemini kullanan anahtar kapsülleme protokolleri arasında FrodoKEM protokolünden sonra uygulama bakımından en iyi durumda olan protokoldür. Lizard protokolünün genel yapısı şu şekildedir (Cheon vd, 2017):

- Tanımlama
    1. Protokol genelinde kullanılan açık parametrelerin belirlenmesi
       Açık parametreler (params): $m, n, \ell_1, \ell_2, \ell, d, p, h_r, \alpha$
       Kullanılan özet fonksiyonlar: G, H, H'
- Anahtar Üretimi (params)
    1. Rastgele $A \in \mathbb{Z}_q^{m \times n}$ ve $T \in \{0,1\}^{\ell_1 \times \ell_2}$ matrisleri üretilir.
    2. Reddetme örneklemesi kullanılarak $S \in \{-1,0,1\}^{n \times \ell_1}$ matrisi üretilir.
    3. CDT ile ayrık Gauss dağılımı kullanılarak $E \in \mathbb{Z}_q^{m \times \ell_1}$ hata matrisi üretilir.
    4. $B = -AS + E$ işlemi ile açık anahtarın bir parçası oluşturulur.

5. Çıktı olarak açık anahtar ($(A \parallel B) \in \mathbb{Z}_q^{m\times(n\times\ell_1)}$) ve gizli anahtar ($(S,T) \in \{-1,0,1\}^{n\times\ell_1} \times \{0,1\}^{\ell_1\times\ell_2}$) verilir.

- Kapsülleme (params, $(A \parallel B)$)
  1. Rastgele $M \in \{0,1\}^{\ell_1\times\ell_2}$ matrisi üretilir.
  2. $R \leftarrow H(M)$ ve $d \leftarrow H'(M)$
  3. $C_1 \leftarrow (p/q) \cdot A^t R \in \mathbb{Z}_p^{n\times\ell_2}$ ve $C_2 \leftarrow (p/q)\cdot((q/2)\cdot M + B^t R \in \mathbb{Z}_p^{\ell_1\times\ell_2}$ şifre metinleri elde edilir.
  4. $K \leftarrow G(C_1, C_2, d, M)$
  5. Çıktı olarak şifre metinler ve oturum anahtarı $(C = (C_1, C_2, d), K)$ verilir.
- Kapsülü Çözme (params, $(A \parallel B)$, (S,T), C)
  1. $(C_1, C_2, d) \leftarrow C$
  2. $M' \leftarrow (2/p)\cdot(C_2 + S^t C_1) \in \mathbb{Z}_2^{\ell_1\times\ell_2}$
  3. $R' \leftarrow H(M')$ ve $d' \leftarrow H'(M')$
  4. $C_1' \leftarrow (p/q) \cdot A^t R' \in \mathbb{Z}_p^{n\times\ell_2}$ ve $C_2' \leftarrow (p/q)\cdot((q/2)\cdot M' + B^t R' \in \mathbb{Z}_p^{\ell_1\times\ell_2}$
  5. $C' \leftarrow (C_1', C_2', d')$
  6. Eğer $C \neq C'$ise $K \leftarrow (C_1, C_2, d, T)$
  7. Aksi halde, $K \leftarrow (C_1, C_2, d, M')$

Lizard protokolünün FrodoKEM protokolüne göre farklılıklarından ilki; LWE örnekleri oluşturulması aşamasında $B \leftarrow AS + E$ işlemi yerine $B \leftarrow -AS + E$ işlemini kullanmasıdır. Bir diğer ve bizim açımızdan en önemli fark matris boyutları ve uygulamada kullanılan çarpım şekilleridir. Örneğin; FrodoKEM protokolünde A matrisi $n \times n$ boyutunda kare şeklinde rastgele bir matris ve S matrisi $n \times \bar{n}$ boyutunda ayrık Gauss dağılımı ile üretilen bir matrisken, Lizard protokolünde ise A matrisi $m \times n$ boyutunda dikdörtgen şeklinde bir rastgele matris ve S matrisi $n \times \ell_1$ boyutunda {-1, 0, 1} elemanlarından oluşan rastgele bir matrisdir (Naehrig vd, 2017; Cheon vd, 2017). Lizard protokolü matris vektör çarpım işlemleri açısından FrodoKEM protokolünden geride kalmaktadır. Bunun sebebi protokol içerisinde standart matris çarpım algoritması (schoolbook) kullanılmasıdır. Standart matris çarpım algoritması uygulamalar üzerinde yavaş çalışmakta olan bir algoritmadır ve bu durum Lizard protokolünün yavaş çalışmasına sebep olmaktadır. Bundan dolayı

Lizard protokolünü daha verimli hale getirebilmek ve hızlandırmak amacıyla protokol içerisindeki matris vektör çarpım işlemleri oluşturulmuş olan kütüphane içerisindeki çarpımlar ile değişitirilerek elde edilen sonuçlar incelenmiştir. Fakat bu çarpımlar eklenirken protokolün uygulamasındaki farklılıklar göz önüne alınmıştır. Protokolün içerisinde bulunmayan transpoz işlemleri uygulamada bulunmaktadır. Örneğin anahtar üretimi kısmında protokolün tanımında tranpoz işlemi yoktur fakat uygulamada S matrisi traspozu alınarak kullanılmaktadır. Bu gibi durumlardan dolayı; bazı çarpım şekilleri için önce gerekli matrislerin transpozları alınması durumu ortaya çıkmaktadır.

Matris vektör çarpım kütüphanesi Lizard protokolüne uygulandığında elde edilen sonuçlar Çizelge 5.2'de verilmiştir. Lizard protokolünün NIST yarışmasında önerilmiş olan toplamda 6 parametre seti içinden, her bir güvenlik seviyesinden bir tane olacak şekilde Lizard-536, Lizard-952 ve Lizard-1300 (AES – 128, AES – 192, AES – 256) parametre setleri üzerinde matris çarpım kütüphanesi uygulaması yapılmıştır.

Çizelge 5.2. Matris vektör çarpım kütüphanesi uygulaması sonrasında Lizard protokolündeki değişimler

	Standart Matris Vektör Çarpımı ile Lizard Protokolünün Alt Algoritmalarının Çalışma Süreleri (ms)			Otomatikleştirilmiş Matris Vektör Çarpımı ile Lizard Protokolünün Alt Algoritmalarının Çalışma Süreleri (ms)		
N	K	E	D	K	E	D
536	2.875	3.120	3.111	1.953	2.521	2.503
952	8.391	5.306	5.436	3.718	4.461	4.631
1300	6.718	7.276	7.398	5.578	6.528	6.428

5.1.1. Lizard protokolünü uygulama ataklarına karşı korumak için yapılan değişiklikler

Yan kanal saldırıları; saldırı yapan tarafın kriptografik sistem hakkında güç tüketimi, cihazın yaydığı ısı, çalışma süresi gibi bilgileri toplayabileceği pasif saldırılardır (Taha ve Eisenbarth, 2015). Bu bilgilerin her biri gizli anahtara bağlı olan modül içerisinde kullanılan verilere bağlı olarak değişmektedir. Yan kanal saldırıları kriptografik sistemler için büyük bir güvenlik tehditi oluşturmaktadır. Çünkü saldırgan gizli anahtarın bütünü yerine, küçük parçalarını (sub-key) elde etmeye çalışabilir. Bundan dolayı anahtar boyutunun artırılması kriptografik sistemleri yan kanal saldırılarına karşı korumakta yeterli olmamaktadır (Taha ve Eisenbarth, 2015). Yazılımlar üzerinde daha farklı önlemler alınması gerekliliği ortaya çıkmaktadır. Bu kısımda Lizard protokolü üzerinde Han ve arkadaşlarının 2018 yılında yaptıkları bir çalışma (Han vd, 2018) incelenmiştir. Lizard[1] protokolünün GitHub üzerinde paylaşılmış olan açık kaynaklı bir uygulaması üzerine yapılan bu çalışmada Han vd. , Bindel ve arkadaşları tarafından 2016 yılında önerilmiş olan yazılımlar üzerindeki hata analizi (Fault Analysis) (Bindel vd, 2016) yöntemlerini kullanarak gizli anahtarı elde edebildiklerini söylemişlerdir (Han vd, 2018). Genellikle yapılanın aksine bu atak yazılım üzerine yapılmıştır. Ek olarak bu ataklara karşı ve uygulamanın genel olarak iyileştirilmesi açısından bazı önerilerde bulunmuşlardır. Uygulamada yapılan değişiklikler bu çalışmada önerilenler doğrultusunda yapılmıştır.

Uygulama üzerinde yapılan ilk değişiklik rastgele sayı üretim fonksiyonlarının değiştirilmesidir. Lizard protokolünün bu uygulamasında rastgele sayı üretimi için C programlama dilinin rastgele sayı üretici fonksiyonu rand() kullanılmıştır. Bu fonksiyonun güvensiz olduğuna dair tam bir kanıt olmasada, kriptografik sistemlerin güvenliği açısından kullanılması tercih edilmeyen bir fonksiyondur. rand() fonksiyonu belirli bir sayıda çalıştırıldıktan sonra sürekli aynı çıktıyı vermeye başlamaktadır (Han vd, 2018). Bunun sebebi bu fonksiyonun girdi olarak sistem saatini kullanmasıdır ve bu girdinin entropisi düşüktür. Rastgele sayı üretirken belirli bir sayıya göre üretim yapmak güvenliği artırsa da ve bu sayı her saniyede değişiyor olsa bile kriptografik protokoller için bu güvenlik seviyesi yeterli olmamaktadır. Han vd. 2018 yılında yaptıkları çalışmada rand() fonksiyonu yerine biraz daha güvenli olan dev/urandom fonksiyonunun kullanılmasını önermişlerdir. Fakat AES veya

[1] https://github.com/LizardOpenSource/Lizard_c

cSHAKE gibi yöntemler ile rastgele sayı üretmek, rand() veya dev/urandom fonksiyonları ile rastgele sayı üretmekten çok daha güvenli olmaktadır. Bu durumdan dolayı protokol içerisindeki bütün rand() fonksiyonları cSHAKE-128 fonksiyonu ile değiştirilmiştir. Ek olarak ayrık Gauss dağılımı ile örnek üretimi aşamasında kullanılan rand() fonksiyonları uygulamadan çıkarılmıştır. Bu sayede rastgele sayı üretimi özelinde ugyulama daha güvenli hale gelmiştir.

Uygulama üzerinde yapılan ikinci değişiklik ise; *S* matrisinin üretildiği gen_sk_CPA() fonksiyonunda yapılmıştır. *S* matrisi {-1, 0 ,1} sayılarından oluşmakta olan rastgele bir matrisdir. *S* matrisi üretilirken reddetme örneklemesi kullanılmaktadır. Bu matrisin üretimi sırasında -1 ve 1 değerleri %25 ihtimalle 0 değeri ise %50 ihtimal ile seçilmektedir (Han vd, 2018, Cheon vd, 2018). Han vd. yaptıkları çalışmada, tutarsızlığın giderilmesi için, bu ihtimallerin bütün değerler için eşit (%33) olacak şekilde değiştirilmesi gerektiğini söylemişlerdir. Protokolde *S* matrisi üretilirken kullanılan reddetme örneklemesi kodları üzerinde -1, 0 ve 1 değerlerinin seçilme (reddedilme) ihtimalleri eşit olacak şekilde düzenlemesi yapılmıştır. gen_sk_CPA() fonksyionu üzerinde yapılan değişiklikler Şekil 5.1'de gösterilmiştir.

```
void gen_sk_CPA() {


    /*for (int i = 0; i < LWE_L; ++i) {
        uint16_t* sk_i = sk_CPA + LWE_N * i;
        for (int j = 0; j < LWE_N; ++j) {
            sk_i[j] = (rand() & 0x01) + (rand() & 0x01) - 1;
        }
    }*/

    int count = 0, dmsp = 0;
    unsigned char seed_A[32] = { 0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7,0,1,2,3,4,5,6,7 };
    uint16_t* sk_i = sk_CPA;
    cshake128_simple((uint8_t*)sk_i, LWE_N*LWE_L * 2, 0, seed_A, 32);
    SAMPLE_DG(sk_i, LWE_N*LWE_L);
    for (int i = 0; i < LWE_N*LWE_L; i++) {
        count = 0;
        if ((sk_i[i] & 0x03) != 1) {
            sk_i[i] = (sk_i[i] & 0x01) + ((sk_i[i] >> 1) & 0x01) - 1;
        }
        else if ((sk_i[i] & 0x03) == 1) {
            if (count < 8) {
                sk_i[i] = (sk_i[i] >> 2);
                count++;
            }
            else {
                count = 0;
                cshake128_simple(sk_i + i, 2, dmsp++, seed_A, 32);
            }
        }
    }
}
```

Şekil 5.1. Lizard_c protokolünde gen_sk_CPA() fonksiyonu üzerinde yapılan değişiklikler. Yorum satırı içerisindeki kodlar protokolün orijinal kodlarıdır.

5.2. Emblem Protokolünde Yapılan Değişiklikler ve Elde Edilen Sonuçlar

Emblem protokolü NIST standartlaştırma sürecinde önerilmiş olan anahtar kapsülleme protokollerinden bir diğeridir. Emblem protokolü tez kapsamında incelenmiş olan protokoller arasından kullandığı rastgele sayı üretme fonksiyonlarından dolayı hem çalışma süresi hem de güvenlik açısından en zayıf durumda olan protokoldür. Emblem protokolü, açık anahtarlı şifreleme sistemi olan Emblem.CPA protokolü üzerinden türetilmiş, IND-CCA güvenliğe sahip olan bir anahtar kapsülleme protokolüdür (Seo vd, 2017). Emblem.CPA protokolü içerisindeki şifreleme ve şifre çözme fonksiyonları kullanılmaktadır. Emblem protokolünün genel yapısı şu şekildedir (Seo vd, 2017);

- Tanımlama
    1. Protokol genelinde kullanılan açık parametrelerin belirlenmesi

       Açık parametreler (params) : $m, n, k, t, v, s, q, \sigma, B < \sigma$

       Kullanılan özet fonksiyonlar: $G, H, \widehat{H}$
- Anahtar Üretimi (params)
    1. Rastgele $A \in \mathbb{Z}_q^{m \times n}$ matrisi üretilir.
    2. Rastgele $S \in [-B, B]^{n \times k}$ matrisi seçilir.
    3. Gauss dağılımı ile $E \leftarrow \mathbb{Z}_s^{m \times k}$ hata matrisi üretilir.
    4. $B = AS + E$ işlemi ile açık anahtarın bir parçası oluşturulur.
    5. Açık anahtar (pk = (A,B)) ve gizli anahtar (sk = S) çıktı olarak verilir.
- Kapsülleme (params)
    1. $\delta \leftarrow \{0,1\}^{256}, r \leftarrow G(\delta)$
    2. $R \in [-B, B]^{m \times v}$ ve $(E_1, E_2) \in \mathbb{Z}_s^{v \times (n+k)}$matrisleri üretilir.
    3. $M \leftarrow kod$çözücü$(msg \in \mathcal{M}, t, q)$
    4. $C_1 = (C', C'') \leftarrow (R^T A + E_1, R^T B + E_2 + M)$, $C_2 \leftarrow \widehat{H}(\delta)$
    5. $K = H(\delta, C_1, C_2)$
    6. Çıktı olarak sifre metin $C = (C_1, C_2) \in \mathbb{Z}_q^{v \times (n+k)} \times \{0,1\}^{256}$ ve oturum anahtarı $K \in \{0,1\}^{256}$ verilir.
- Kapsülü Çözme (params, C, S)
    1. Şifre metin, C_1 ve C_2 parçalarına ayrılır.

2. $M \leftarrow C_2 - C_1 S$ ve $r \leftarrow G(\delta)$
3. $c_1 = (c', c'') \leftarrow (R^T A + E_1, R^T B + E_2 + M)$, $c_2 \leftarrow \widehat{H}(\delta)$
4. Eğer $c_1 \neq C_1$ veya $c_2 \neq C_2$ ise çıktı olarak hata ($\perp$) döndürülür.
5. Aksi halde çıktı olarak oturum anahtarı $K(\delta, C_1, C_2)$ döndürülür.

Emblem protokolünün FrodoKEM protokolünden farklarından birisi; A matrisinin şeklidir. FrodoKEM protokolünde A matrisi $n \times n$ şeklinde kare bir matris iken, Emblem protokolünde A matrisi $m \times n$ şeklinde dikdörtgen bir matrisdir. Bir diğer ve en önemli fark ise matris vektör çarpımlarıdır. Emblem protokolünde matrisleri çarpmak için standart matris çarpım algoritması kullanılmaktadır ve diğer protokollerde olduğu gibi bu durum protokolün yavaş çalışmasına sebep olmaktadır. Emblem protokolü üzerinde iki aşamalı bir değişim yapılmıştır. İlk olarak matris vektör çarpım kütüphanesi üzerinden protokole çarpımlar entegre edilmiştir. Fakat protokolde herhangi bir hızlanma veya iyileşme olmamıştır. Bunun sebebi protokol içerisinde bulunan rastgele sayı üretme fonksiyonlarıdır. Emblem protokolünde rastgele sayı üretmek için C programlama dilinin rastgele sayı üretme fonksiyonu "rand()" kullanılmaktadır. Fakat bu fonksiyon protokolü yavaşlatmakla beraber güvenliğinide büyük ölçüde azaltmaktadır. Bir önceki bölümde anlatıldığı gibi rand() fonksiyonu kriptografik sistemler için güvensiz bir fonksiyondur ve protokollerin güvenliği için kullanılması önerilmemektedir. Ek olarak ayrık Gauss dağılımı ile örnekleme yapma aşamasında da Emblem protokolünde yine rand() fonksiyonu kullanılmaktadır. Protokol içerisinde bu rastgele sayı üretme işlemleri daha güvenli ve düzgün olacak şekilde değiştirilmiştir. Anahtar üretimi kısmında A ve S matrislerinin üretimi AES algoritması kullanılacak şekilde değiştirilmiştir. Ayrık Gauss dağılımı kullanılan kısımlarda ise rand() fonksiyonları uygulamadan çıkarılmıştır. Bu rastgele sayı üretim fonksiyonları ile protokolün güvenliğinin, matris vektör çarpımları ile de protokolün verimliliğinin artırılması hedeflenmiştir. Yapılan işlemler sonrasında Emblem protokolünün çalışma hızındaki değişimler Çizelge 5.3'de verilmiştir. Emblem protokolünün her bir parametre şeklinden (611 ve 770) birinci, üçüncü ve beşinci parametre setlerine bu uygulamalar yapılmıştır. Protokolün önceki haline göre anahtar üretiminde yavaşlama, kapsülleme ve kapsülü çözme kısımlarında ise büyük oranda hızlanmalar olmuştur. Anahtar üretimindeki yavaşlamanın sebebi ise

rastgele sayı üretmek için AES algoritmasının kullanılmasıdır. AES algoritmasının maliyeti rand() fonksiyonu ile rastgele sayı üretimine göre çok daha yüksek olduğundan bu kısımda yavaşlama olmuştur. Fakat uygulamanın güvenliği öncekine haline oranla artmıştır. Ek olarak; önceki bölümlerde bahsedildiği gibi AES algoritması 128 bitlik girdiler ile çalıştığından (16 byte) oluşturulacak matrisin boyutları 16'nın katı olacak şekilde genişletilmiştir.

Çizelge 5.3. Matris vektör çarpım kütüphanesi ve rastgele sayı üretimi değişiklikleri sonrasında Emblem protokolündeki değişimler

	Standart Matris Vektör Çarpımı ve Güvensiz Rastgele Sayı Üretimi ile Emblem Protokolünün Alt Algoritmalarının Çalışma Süreleri (ms)			Otomatikleştirilmiş Matris Vektör Çarpımı ve Güvenli Rastgele Sayı Üretimi ile Emblem Protokolünün Alt Algoritmalarının Çalışma Süreleri (ms)		
N	K	E	D	K	E	D
611 (1)	8.625	26.093	24.015	17.291	2.609	2.093
770 (1)	12.140	32.500	32.359	27.406	3.406	3.000
611 (3)	9.078	6.750	6.015	18.359	0.500	0.609
770 (3)	13.718	7.546	8.812	28.046	0.875	0.765
611 (5)	9.390	2.984	2.968	19.359	0.234	0.250
770 (5)	18.421	2.078	2.015	29.500	0.156	0.218

5.3. Lotus Protokolünde Yapılan Değişiklikler ve Elde Edilen Sonuçlar

Lotus protokolünün açık anahtarlı şifreleme şeması şeklinden (LOTUS - PKE), anahtar kapsülleme protokolüne (LOTUS – KEM) dönüştürülmüş versiyonudur. Protokolün yazarları tarafından bu sistem tek kullanımlık şifre üreten ve altında simetrik şifreleme özellikleri bulunan bir sistem olarak tasarlanmıştır (Le Trieu Phong vd, 2017). NIST standartlaştırma süreci kapsamında önerilmiş olan, standart

kafesleri kullanan bir diğer anahtar kapsülleme protokolüdür. Lotus – PKE açık anahtarlı şifreleme sistemindeki fonksiyonları kullanmaktadır. Protokolün genel yapısı şu şekildedir (Le Trieu Phong vd, 2017):

- Tanımlama
    1. Protokol genelinde kullanılan açık parametrelerin belirlenmesi
       Açık parametreler (λ) : $q, n, l, KeyLen$
       Kullanılan özet fonksiyonlar: G, H
- Anahtar Üretimi (λ)
    1. Rastgele $A \in \mathbb{Z}_q^{n \times n}$ matrisi üretilir.
    2. Sapma değeri $s \in \mathbb{R}$ belirlenir.
    3. Ayrık Gauss dağılımı ile $E, S \leftarrow \mathbb{Z}_{(0,s)}^{n \times l}$ matrisleri üretilir.
    4. $B = E - AS \in \mathbb{Z}_q^{n \times l}$ işlemi ile açık anahtarın bir parçası oluşturulur.
    5. Açık anahtar ($pk = B, A, q, n, l, s, KeyLen$) ve gizli anahtar ($sk = S, pk$) çıktı olarak verilir.
- Kapsülleme (pk)
    1. $K \in \{0,1\}^{KeyLen}$ rastgele değeri üretilir.
    2. $\sigma \in \{0,1\}^l$ rastgele değeri seçilir.
    3. $c_{sym} = G(\sigma) \oplus K \in \{0,1\}^{KeyLen}$ ve $h = H(\sigma \parallel c_{sym})$
    4. Gauss hata vektörleri $e_1, e_2 \in \mathbb{Z}_{(0,s)}^{1 \times n}$ ve $e_3 \in \mathbb{Z}_{(0,s)}^{1 \times l}$ üretilir.
    5. $Enc_{pk}^{cpa}(\sigma; h) \rightarrow c_1 = e_1 A + e_2 \in \mathbb{Z}_q^{1 \times n}$, $c_2 = e_1 B + e_3 + \sigma \cdot \lfloor q/2 \rfloor \in \mathbb{Z}_q^{1 \times l}$ işlemleri ile şifre metin parçaları üretilir.
    6. Çıktı olarak $CT = (c_1, c_2, c_{sym})$ şifre metni ve K oturum anahtarı gönderilir.
- Kapsülü Çözme (sk, CT)
    1. $\bar{\sigma} = c_1 + c_2 \in \mathbb{Z}_q^l$
    2. $\bar{\sigma} = (\bar{\sigma}_1, \dots, \bar{\sigma}_2)$. $Eğer\ \bar{\sigma}_i \in [-\lfloor q/4 \rfloor, \lfloor q/4 \rfloor] \subset \mathbb{Z}_q$ ise $\sigma_i' = 0$, aksi halde $\sigma_i' = 1$
    3. $\sigma' = \sigma', \dots, \sigma_l'$ ve $h' = H(\sigma' \parallel c_{sym})$
    4. $Enc_{pk}^{cpa}(\sigma'; h') \rightarrow c_1' = e_1' A + e_2' \in \mathbb{Z}_q^{1 \times n}$, $c_2' = e_1' B + e_3' + \sigma' \cdot \lfloor q/2 \rfloor \in \mathbb{Z}_q^{1 \times l}$ işlemleri ile şifre metin parçaları tekrar üretilir.

5. Eğer $(c_1, c_2) \neq (c_1', c_2')$ ise çıktı olarak hata ($\perp$) döndürülür.
6. Aksi halde K oturum anahtarı paylaşılır.

Lotus protokolünün FrodoKEM protokolüne göre farklılıklarından birisi LWE örneklerinin üretimi aşamasında yapılan işlemdir. Bu işlem; FrodoKEM protokolünde $AS + E$ şeklinde iken, Lotus protokolünde $-AS + E$ şeklindedir. Ayrık Gauss dağılımı ile hata örneklemesi aşamasında ise FrodoKEM den farklı olarak Knuth-Yao yöntemi kullanılmaktadır. Tez kapsamında asıl ilgilenilen farklılık ise matris vektör çarpımlarıdır. FrodoKEM dışındaki diğer protokollerde olduğu gibi Lotus protokolünde matris vektör çarpımı için standart matris çarpım algoritması kullanılmaktadır. Protokolde bu kısmın verimliliğinin artırılması için oluşturulmuş olan matris vektör çarpım kütüphanesinden $A * S$ çarpımı protokole eklenmiştir. Fakat protokolün uygulamasında LWE örneklerinin üretilmesi aşamasındaki farklılıklar göz önüne alınmıştır. Bu farklılıklardan ilki $A * S$ işlemi yerine $-A * S$ işlemi kullanılmasıdır. İkinci farklılık ise Lotus protokolünde S matrisinin transpozunun alınarak çarpım işleminin yapılmasıdır. Kütüphane içerisindeki $A * S$ çarpımına, protokolün düzgün çalışması için bu gibi değişiklikler eklenerek çarpım işlemi gerçekleştirilmiştir. Diğer sistemlere göre ek olarak çarpım işleminden önce S matrisinin transpozu alınarak işleme başlanmıştır. Bu durum çarpım işleminin yavaşlamasına sebep olan bir durumdur. Fakat parametre setlerine göre ayrı ayrı incelendiğinde kütüphane içersindeki çarpım işlemi standart matris çarpım algoritmasına göre daha verimli çalışmaktadır. Protokol içerisinde sadece anahtar üretimi kısımda matris vektör çarpım işlemi bulunmaktadır. Lotus protokolünün üç farklı parametre seti için anahtar üretimi kısmındaki çarpım işlemi değiştirilip incelemesi yapılmıştır. Elde edilen sonuçlar sadece anahtar üretimi açısından incelenmiştir. Bunun dışında Lotus protokolü üzerinde başka bir değişiklik yapılmamıştır. Yapılan işlemler protokolün referans uygulaması üzerinde yapılmıştır. Elde edilen sonuçlar Çizelge 5.4'de verilmiştir.

Çizelge 5.4. Anahtar üretimi kısmına matris vektör çarpım kütüphanesi uygulaması sonrasında Lotus protokolündeki değişimler

N	Standart Matris Vektör Çarpımı ile Lotus Protokolünün Anahtar Üretim Algoritmasının Çalışma Süreleri (ms)	Otomatikleştirilmiş Matris Vektör Çarpımı ile Lotus Protokolünün Anahtar Üretim Algoritmasının Çalışma Süreleri (ms)
576	8.105	3.466
704	14.593	9.004
832	21.611	15.445

5.4. Oluşturulmak İstenen LWE Yazılım Kütüphanesi İçin Yapılanlar

Kuantum sonrası kriptografi için yapılan NIST standartlaştırma süreci kapsamında önerilen protokollerin tek bir çatıda toplanması için FrodoKEM protokolü temel olarak kullanılmıştır. Bunun için FrodoKEM protokolü baştan oluşturulmuştur ve öncesinde yapılmış olan matris vektör çarpım kütüphanesi sisteme eklenmiştir. Uygulama yapılırken bütün seçimlerin kullanıcıya bırakılması hedeflenmiştir. Örneğin kullanıcı parametreleri belirledikten sonra, kullanacağı rastgele sayı üretim fonksiyonunu veya kullanmak istediği matris çarpım yöntemini seçebilmektedir. Rastgele sayı üretimi için AES veya cSHAKE algoritması kullanılabilmektedir. Bu uygulama yapılırken matris vektör çarpımı aşamasında FrodoKEM' deki yöntem tam anlamıyla kullanılmıştır. Kütüphane içerisinde matris çarpımından önce matrislerin tamamı üretilip sonrasında matris içinden istenilen kısım kullanılmaktaydı. Bu kısımda ise kullanılacak çarpım yöntemine göre matrisin tamamı değil bir kısmı üretilmektedir. $A * S$ çarpımı için; A matrisinin tamamı yerine, her seferinde dört satırı üretilip S matrisi ile çarpılmaktadır. $S' * A$ çarpımı için; yine A matrisinin tamamı yerine, her seferinde sekiz sütun üretilip S' matrisi ile çarpılmaktadır. FrodoKEM önceden A matrisinin kare matris, $640 \times 640, 976 \times 976$ ve 1344×1344 boyutlarında olduğu durumlarda çalışmaktayken, A matrisinin boyutu ve şekli fark etmeksizin çalışır hale gelmiştir. Bu şekilde diğer protokollerin bu sisteme uyarlanabilmesi için temel oluşturulmuştur.

6. SONUÇ VE GELECEK ÇALIŞMALAR

Bu tez kapsamında kuantum sonrası kriptografi için zor problemler arasında bulunan LWE problemini temel alan anahtar kapsülleme protokolleri incelenmiştir. LWE problemi içerisinde matris vektör çarpım işlemi bulundurmaktadır ve bu çarpım işlemi uygulamalarda rastgele sayı üretiminden sonra en çok vakit almakta olan kısım olarak dikkat çekmektedir. İncelenen protokoller arasında FrodoKEM protokolü tez kapsamında temel alınmıştır ve bu protokolde kullanılan yöntemler, iyileştirmeler incelenmiştir. Bu doğrultuda FrodoKEM protokolü içerisindeki matris vektör çarpım işlemleri temel alınarak, kafes tabanlı sistemlerde kullanılması amacıyla bir matris vektör çarpım kütüphanesi oluşturulmuştur. Bu kütüphane içerisindeki çarpım işlemleri FrodoKEM protokolündeki çarpım yöntemlerinin genelleştirilmiş versiyonlarıdır.

Sonraki aşamada Lizard, Emblem ve Lotus protokolleri detaylı olarak incelenmiş ve yapılmasının gerekli olduğu düşünülen iyileştirmeler belirlenmiştir. Matris vektör çarpım kütüphanesi içerisindeki çarpımlar, bu protokollerde kullanılan standart matris çarpım algoritması yerine kullanılarak protokollerdeki değişimler incelenmiştir. Tez kapsamında FrodoKEM dışındaki protokollerin, matris vektör çarpımlarına ek olarak protokollerin genel verimliliklerinin artırılmasıda hedeflenmiştir. FrodoKEM temel alınarak oluşturulmuş olan matris vektör çarpım işlemlerinin, standart matris çarpım algoritmasına göre uygulamalarda daha verimli olduğu gözlemlenmiştir. Sonuçlar elde edilirken Intel Core i7-4700HQ 2.40GHz işlemci ve 16 GB RAM'a sahip bir bilgisayar kullanılmıştır. Kodların derlenmesi ve çalıştırılması için gcc 5.4.0 derleyicisi ve derleyici tarafından sağlanan en iyi optimizasyon seçeneği kullanılmıştır (-03). Bütün kodlamalar C programlama dilinde yapılmıştır. Tez kapsamında yazılmış ve kullanılmış olan bütün kodlar https://github.com/Justice0893 adresinde paylaşılmıştır.

Gelecek çalışma olarak; matris vektör çarpım kütüphanesinin AVX2 ve AVX-512 komut setleri kullanılarak genişletilmesi ve diğer protokollere uygulanması üzerine düşünülmektedir. Ek olarak oluşturulmak istenen LWE yazılım kütüphanesinin tamamlanması hedeflenmektedir.

KAYNAKLAR

Ajtai, M. 1996. Generating hard instances of lattice problems. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, ACM, 99-108.

Ajtai, M. and Dwork, C. 1997. A public-key cryptosystem with worst-case/average-case equivalence. In STOC (Vol. 97, pp. 284-293).

Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q. and Perlner, R. 2019. Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process. US Department of Commerce, National Institute of Standards and Technology.

Alkim, E., Bos, J., Ducas, L., Longa, P., Mironov, I., Naehrig, M. and Easterbrook, K. 2019. FrodoKEM: Learning With Errors key encapsulation.

Bernstein, D. J., Buchmann, J. and Dahmen, E. Post-Quantum Cryptography. 2009. Springer, Berlin.

Bernstein, E. and Vazirani, U. 1997. Quantum complexity theory. *SIAM Journal on computing*, *26*(5), 1411-1473.

Bindel, N., Buchmann, J. and Krämer, J. 2016. Lattice-based signature schemes and their sensitivity to fault attacks. In *2016 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)* (pp. 63-77). IEEE.

Bogdanov, D. 2005. MTAT. 07.006 Research Seminar in Cryptography IND-CCA2 secure cryptosystems.

Bone, S. and Castro, M. 1997. A brief history of quantum computing. *Imperial College in London, http://www. doc. ic. ac. uk/~ nd/surprise_97/journal/vol4/spb3*.

Boneh, D. and Shoup, V. 2015. A graduate course in applied cryptography. *Draft 0.2*.

Bos, J., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V. and Stebila, D. 2016. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 1006-1018). ACM.

Brassard, G., Høyer, P. and Tapp, A. 1998. Quantum cryptanalysis of hash and claw-free functions. In Latin American Symposium on Theoretical Informatics (pp. 163-169). Springer, Berlin, Heidelberg.

Buchanan, W. and Woodward, A. 2017. Will quantum computers be the end of public key encryption?. *Journal of Cyber Security Technology*, *1*(1), 1-22.

Chen, L., Jordan, S., Liu, Y-K., Moody, D., Peralta, R., Perlner, R. and Smith-Tone, D. 2016. NISTIR 8105, Report on Post-Quantum Cryptography, NIST. National Institute of Standards and Technology.

Cheon, J. H., Kim, D., Lee, J. and Song, Y. 2017. *Lizard Public Key Encryption*. Technical report, National Institute of Standards and Technology, 2017 available at https://csrc. nist. gov/projects/post-quantum-cryptography/round-1-submissions, Erişim Tarihi: 14.05.2019.

Cheon, J. H., Kim, D., Lee, J. and Song, Y. 2018. Lizard: Cut Off the Tail! A Practical Post-quantum Public-Key Encryption from LWE and LWR. In International Conference on Security and Cryptography for Networks (pp. 160-177). Springer, Cham.

Coretti, S., Maurer, U. and Tackmann, B. 2013. A Constructive Perspective on Key Encapsulation. In *Number Theory and Cryptography* (pp. 226-239). Springer, Berlin, Heidelberg.

Datta, A., Derek, A., Mitchell, J. C. and Warinschi, B. 2006. Key Exchange Protocols: Security Definition, Proof Method and Applications. *IACR Cryptology ePrint Archive*, *2006*, 56.

Diffie, W. and Hellman, M. 1976. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6): 644-654

Diffie, W., Van Oorschot, P. C. and Wiener, M. J. 1992. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, *2*(2), 107-125.

Ding, J. and Schmidt, D. 2005. Rainbow, a new multivariable polynomial signature scheme. In International Conference on Applied Cryptography and Network Security (pp. 164-175). Springer, Berlin, Heidelberg.

Ding, J. and Yang, B. Y. 2009. Multivariate public key cryptography. In *Post-quantum cryptography* (pp. 193-241). Springer, Berlin, Heidelberg.

Ducas, L., Durmus, A., Lepoint, T. and Lyubashevsky, V. 2013. Lattice signatures and bimodal Gaussians. In *Annual Cryptology Conference* (pp. 40-56). Springer, Berlin, Heidelberg.

ElGamal, T. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4): 469-472

FIPS 197. 2001. FIPS 197 Advanced encryption standard (AES). Federal information processing standards publication, National Institute of Standards and Technology, 197(441), 0311.

Flatten 2D matrix. C++ uses row major order: N x m, which are the number of rows and columns also called the height and the width - PDF. https://docplayer.net/51943838-Flatten-2d-matrix-c-uses-row-major-order-n-x-m-which-are-the-number-of-rows-and-columns-also-called-the-height-and-the-width.html (Erişim tarihi: 21.Ağustos.2019)

Fujisaki, E. and Okamoto, T. 1999. Secure integration of asymmetric and symmetric encryption schemes. In Annual International Cryptology Conference (pp. 537-554). Springer, Berlin, Heidelberg.

Galbraith, S. D. 1999. Constructing isogenies between elliptic curves over finite fields. LMS Journal of Computation and Mathematics, 2, 118-138.

Gentry, C., Peikert, C. and Vaikuntanathan, V. 2008. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing* (pp. 197-206). ACM.

Goldreich, O., Goldwasser, S. and Halevi, S. 1997. Public-key cryptosystems from lattice reduction problems. In Annual International Cryptology Conference (pp. 112-131). Springer, Berlin, Heidelberg.

Grover, L. K. 1996. A fast quantum mechanical algorithm for database search. Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, ACM, 212-219.

Han, S., Choi, N., An, H., Choi, R. and Kim, K. 2018. Prey on Lizard: Mining Secret Key on Lattice-based Cryptosystem. In *2018 Symposium on Cryptography and Information Security (SCIS 2018)*. IEICE Technical Committee on Information Security.

Hirschhorn, P. S., Hoffstein, J., Howgrave-Graham, N. and Whyte, W. 2009. Choosing NTRUEncrypt parameters in light of combined lattice reduction and

MITM approaches. In International Conference on Applied Cryptography and Network Security (pp. 437-455). Springer, Berlin, Heidelberg.
Hoffstein, J., Lieman, D., Pipher, J. and Silverman, J. H. 1999. NTRU: A public key cryptosystem. *Submissions and Contributions to IEEE P*, 1363
Hoffstein, J., Pipher, J. C., Silverman, J. H. and Silverman, J. H. 2008. *An introduction to mathematical cryptography*. Springer,
Howe, J. and O'Neill, M. 2017. GLITCH: A Discrete Gaussian Testing Suite For Lattice-Based Cryptography. *IACR Cryptology ePrint Archive*, *2017*, 438.
Howe, J., Khalid, A., Rafferty, C., Regazzoni, F. and O'Neil, M. 2016. On practical discrete Gaussian samplers for lattice-based cryptography. *IEEE Transactions on Computers*, *67*(3), 322-334.
Howe, J., Oder, T., Krausz, M. and Güneysu, T. 2018. Standard Lattice-Based Key Encapsulation on Embedded Devices. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 372-393.
Jiang, H., Zhang, Z., Chen, L., Wang, H. and Ma, Z. 2018. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In *Annual International Cryptology Conference* (pp. 96-125). Springer, Cham.
Koblitz, N. 1987. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177): 203-209
Le Trieu Phong, T. H., Aono, Y. and Moriai, S. 2017. Lotus. Technical report, National Institute of Standards and Technology.
Lebeck, A. R. and Wood, D. A. 1994. Cache profiling and the SPEC benchmarks: A case study. *Computer*, *27*(10), 15-26.
Lindner, R. and Peikert, C. 2011. Better key sizes (and attacks) for LWE-based encryption. In Cryptographers' Track at the RSA Conference (pp. 319-339). Springer, Berlin, Heidelberg.
Lomont, C. 2011. Introduction to intel advanced vector extensions. *Intel White Paper*, 1-21.
Mano, M. M. 2005. *Computer system architecture*. Dorling Kindesley Pearson.
Mavroeidis, V., Vishi, K., Zych, M. D. and Jøsang, A. 2018. The impact of quantum computing on present cryptography. arXiv preprint arXiv:1804.00200.
McEliece, R. J. 1978. A public-key cryptosystem based on algebraic. *Coding Thv*, *4244*, 114-116.
Miller,V. S. 1985. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques* (pp. 417-426). Springer, Berlin, Heidelberg.
Moody, D. 2017. The ship has sailed: the NIST post – quantum cryptography competition. 13 Aralık 2018 tarihinde https://cscr.nist.gov/Presentations/2017/The-Ship-Has-Sailed-The-NIST-Post-Quantum-Cryptog sitesinden erişildi.
Naehrig, M., Alkim, E., Bos, J., Ducas, L., Easterbrook, K., LaMacchia, B., and Raghunathan, A. 2017. FrodoKEM: practical quantum-secure key encapsulation from generic lattices. NIST submissions.
Nejatollahi, H., Cammarota, R., Regazzoni, F., Ray, S., Banerjee, I. and Dutt, N. 2017. Software and Hardware Implementation of Lattice-based Cryptography Schemes.
Nguyen, P. 1999. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from crypto'97. In Annual International Cryptology Conference (pp. 288-304). Springer, Berlin, Heidelberg.

Nguyen, P. and Stern, J. 1998. Cryptanalysis of the Ajtai-Dwork cryptosystem. In Annual International Cryptology Conference (pp. 223-242). Springer, Berlin, Heidelberg.

NIST. 2018. Post Quantum Cryptography. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions (Erişim Tarihi 10.05.2019)

Nielsen, M. A. and Chuang, I. L. 2000. Quantum computation and quantum information.

Peikert, C. 2009. Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual ACM symposium on Theory of computing* (pp. 333-342). ACM.

Peikert, C. 2010. An efficient and parallel Gaussian sampler for lattices. In *Annual Cryptology Conference* (pp. 80-97). Springer, Berlin, Heidelberg.

Peikert, C. 2014. Lattice cryptography for the internet. In *international workshop on post-quantum cryptography* (pp. 197-219). Springer, Cham.

Peikert, C. 2016. A decade of lattice cryptography. *Foundations and Trends® in Theoretical Computer Science*, *10*(4), 283-424.

Pietrzak, K. 2012. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science* (pp. 99-114). Springer, Berlin, Heidelberg.

Regev, O. 2005. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing, 84–93. https://doi.org/10.1145/1060590.1060603

Regev, O. 2006. Lattice-based cryptography. In *Annual International Cryptology Conference* (pp. 131-141). Springer, Berlin, Heidelberg.

Regev, O. 2010. The learning with errors problem. *Invited survey in CCC*, *7*.

Rejection Sampling. 2018. Erişim tarihi 07 Ekim 2019, gönderen website: https://untitledtblog.tistory.com/134

Rivest, R. L., Shamir, A. and Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2): 120-126

Row Major Ordering. http://icarus.cs.weber.edu/~dab/cs1410/textbook/7.Arrays/row_major.html (Erişim tarihi: 22.Ağustos.2019)

Saarinen, M. J. O. 2015. Gaussian Sampling Precision and Information Leakage in Lattice Cryptography. *IACR Cryptology ePrint Archive*, *2015*, 953.

SafeCrypto. 2019. https://www.safecrypto.eu/pqclounge/round-2-candidates/ (Erişim tarihi: 10.05.2019)

Seo, M., Park, J. H., Lee, D. H., Kim, S. and Lee, S. J. 2017. Emblem and r. emblem. Technical report, National Institute of Standards and Technology.

Shor, P. W. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium, 124-134. Ieee.

Taha, M. and Eisenbarth, T. 2015. Implementation Attacks on Post-Quantum Cryptographic Schemes. *IACR Cryptology ePrint Archive*, *2015*, 1083.

The significance of SIMD, SSE and AVX: For Robust HPC Development. Erişim tarihi: 18 Eylül 2019, gönderen Scribd website: https://www.scribd.com/document/321776325/3a-SIMD

Thiyagalingam, J., Beckmann, O. and Kelly, P. H. 2003. An exhaustive evaluation of row-major, column-major and Morton layouts for large two-dimensional

arrays. In *Performance Engineering: 19th Annual UK Performance Engineering Workshop* (pp. 340-351).

Yuan, Y., Xiao, J., Fukushima, K., Kiyomoto, S. and Takagi, T. 2018. Portable Implementation of Postquantum Encryption Schemes and Key Exchange Protocols on JavaScript-Enabled Platforms. *Security and Communication Networks, 2018*.

ÖZGEÇMİŞ

Adı Soyadı : Bilge Kağan Yazar

Doğum Yeri : Ankara

Doğum Tarihi : 19.11.1993

Yabancı Dili : İngilizce

Eğitim Durumu

Lise : Halis Gülle Anadolu Lisesi (2011)

Lisans : Ankara Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği (2017)

Çalıştığı Kurum/Yıl

Ondokuz Mayıs Üniversitesi Bilgisayar Mühendisliği Bölümü Araştırma Görevlisi / Mart 2019 – Halen

Yayınlar

Alkım, E. ve Yazar, B. K. 2019. Post Quantum Learning With Errors Problem Based Key Encapsulation Protocols and Matrix Vector Product. 2019 4th International Conference on Computer Science and Engineering (UBMK), 301-306. https://doi.org/10.1109/UBMK.2019.8907201